

Reverse Engineering - 2

2021/11/26

Presented by LJP

whoami

- LJP / LJP-TW
- SQLab @ NYCU 碩一
- CTF @ 10sec / TSJ
- Pwner



Outline

- Tools
- Where to start
- Address Space
- Speed-up
- DLL
- PE Format
- IAT
- EAT
- PEB

Tools

PE-bear / x64dbg

Tools - PE-Bear

- 觀察 PE format

PE-bear v0.5.4 [C:/Users/pt/Downloads/windows/ntdll.dll]

File Settings View Compare Info

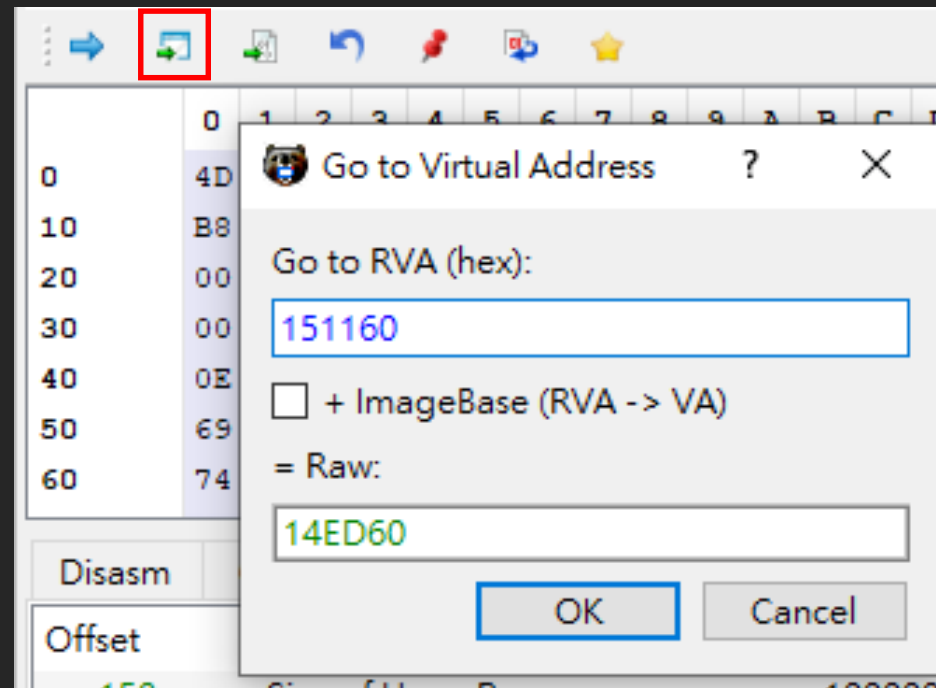
ntdll.dll*

- DOS Header
 - EP = 0
- DOS stub
- NT Headers
 - Signature
 - File Header
 - Optional Header
 - Section Headers
- Sections
 - .text
 - PAGE
 - RT
 - .rdata
 - .data
 - .pdata
 - .mrdata
 - .00cfg
 - .rsrc
 - .reloc
 - Overlay

Offset	Name	Value	Meaning
EC	Machine	8664	AMD64 (K8)
EE	Sections Count	a	10
F0	Time Date Stamp	bd2c3c23	星期一, 28.07.2070 17:06:43 UTC
F4	Ptr to Symbol Table	0	0
F8	Num. of Symbols	0	0
FC	Size of OptionalHeader	f0	240
FE	Characteristics	2022	
		2	File is executable (i.e. no unresolved external r...
		20	App can handle >2gb addresses
		2000	File is a DLL.

Tools - PE-Bear

- 換算 RVA <-> Raw address



Tools - x64dbg

hello.exe - PID: 3988 - 模組: ntdll.dll - 執行緒: 主執行緒 6808 - x64dbg
檔案(F) 檢視(V) 除錯(D) 追蹤(N) 外掛程式(P) 最愛(L) 選項(O) 幫助(H) Jul 1 2021 (TitanEngine)

CPU 日誌 筆記 中斷點 記憶體映射 呼叫堆疊 SEH鏈 腳本 符號 <> 原始碼 引用 執行緒 Handles 追蹤

暫存器

組合語言

資料視窗

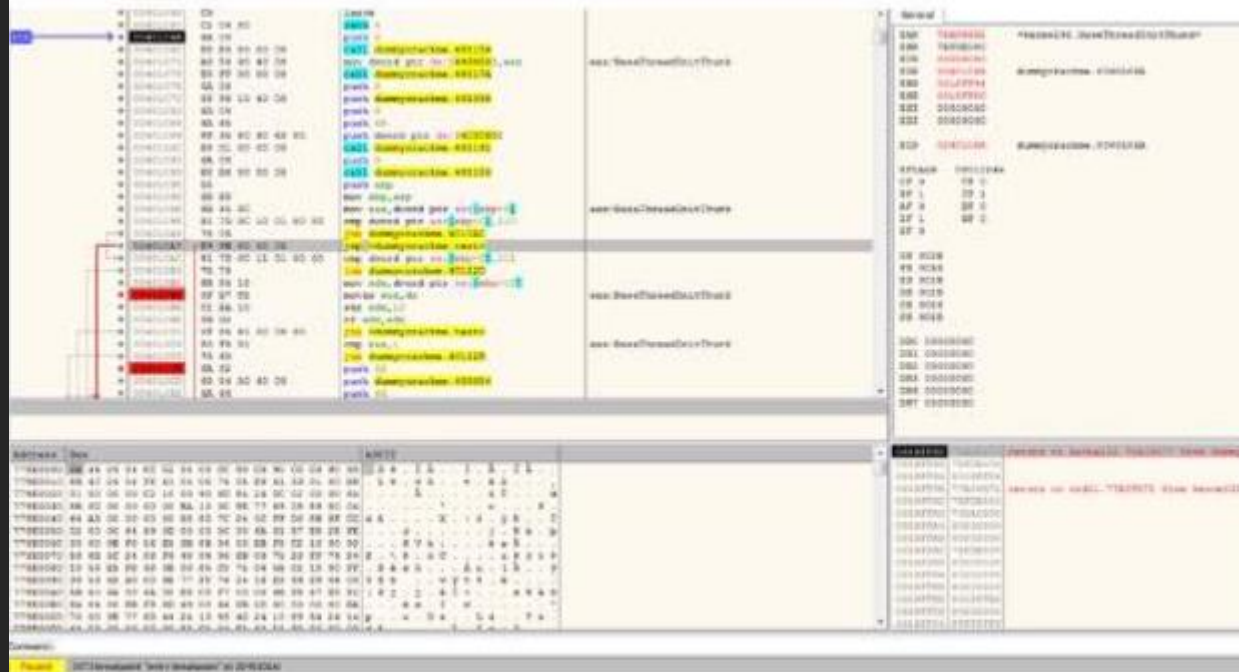
Stack

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

已花費時間: 00:04:22

Tools - x64dbg

Q: 請問附圖 x64dbg 不包含以下哪個介面？



A: Hot keys bar

Tools - x64dbg

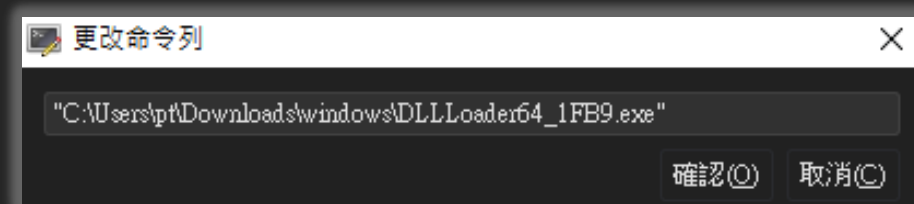


Tools - x64dbg

- 常用快捷鍵
 - F2: 設定中斷點
 - F9: 繼續執行
 - F8: 步過
 - F7: 步入
 - Ctrl+F9: 執行到 ret
 - Ctrl+G: goto
 - Space: 組譯

Tools - x64dbg

- 常用功能
 - 設定指令列



Tools - x64dbg

- 常用功能
 - 查看載入了哪些 module

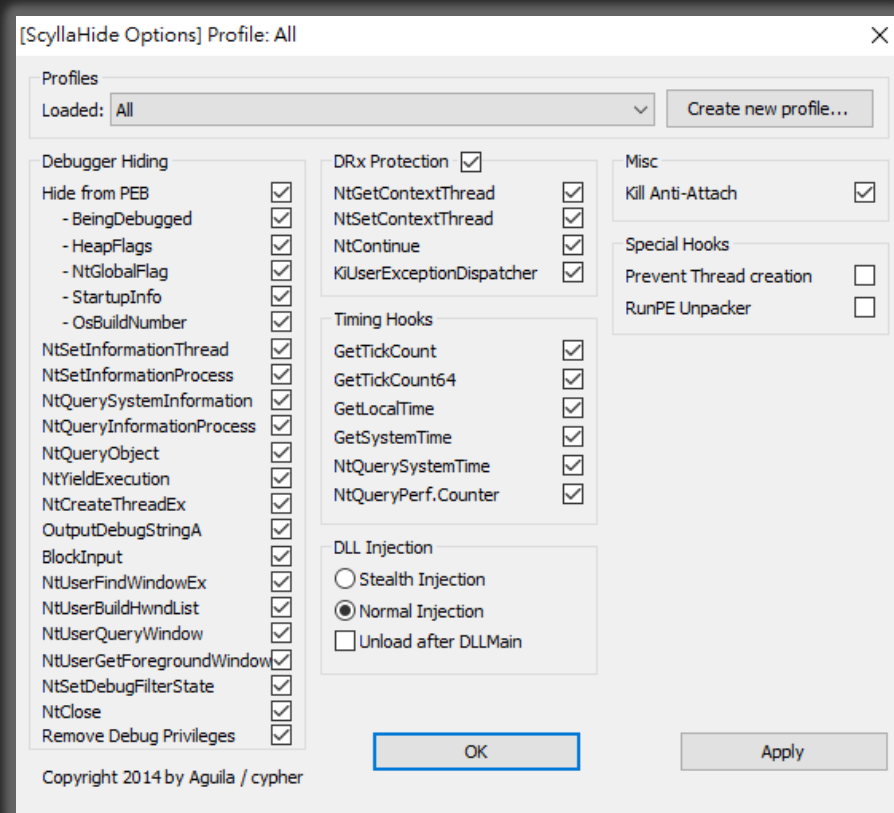
CPU 日誌 筆記 中斷點 記憶體映射 呼叫堆疊 SEH鏈 腳本 符號 原始碼 引用 執行緒 Handles 追蹤										
基址	模組	Party	路徑	Status	位址	類型	序數	符號	符號 (已解碼)	
00007FF739BF0000	dllloader64_1fb9.exe	使用者模組	C:\Users\pt\Downloads\windows\DLLLo	Unloaded	00007FFE1C5FD310	導出	1	??\$_Getvals@_we?\$time_get@DV?\$istr	protected: vo	
00007FFE19E80000	apphelp.dll	系統模組	C:\Windows\System32\apphelp.dll	Unloaded	00007FFE1C5FD310	導出	2	??\$_Getvals@_we?\$time_get@GV?\$istr	protected: vo	
00007FFE1C5D0000	msvc_p_win.dll	系統模組	C:\Windows\System32\msvc_p_win.dll	Unloaded	00007FFE1C5FD310	導出	3	??\$_Getvals@_we?\$time_get@_wv?\$istr	protected: vo	
00007FFE1C6D0000	gdi32full.dll	系統模組	C:\Windows\System32\gdi32full.dll	Unloaded	00007FFE1C618B50	導出	4	??0?\$_Yarne@Dstd@@QEAA@AEBV01@@Z	public: __cde	
00007FFE1C7E0000	kernelbase.dll	系統模組	C:\Windows\System32\kernelBase.dll	Unloaded	00007FFE1C5F57F0	導出	5	??0?\$_Yarne@Dstd@@QEAA@PEBD@Z	public: __cde	
00007FFE1C860000	ucrtbase.dll	系統模組	C:\Windows\System32\ucrtbase.dll	Unloaded	00007FFE1C5E0600	導出	6	??0?\$_Yarne@Dstd@@QEAA@XZ	public: __cde	
00007FFE1CC60000	win32u.dll	系統模組	C:\Windows\System32\win32u.dll	Unloaded	00007FFE1C618B80	導出	7	??0?\$_Yarne@Gstd@@QEAA@AEBV01@@Z	public: __cde	
00007FFE1CDF0000	user32.dll	系統模組	C:\Windows\System32\user32.dll	Unloaded	00007FFE1C618B80	導出	8	??0?\$_Yarne@Gstd@@QEAA@PEBG@Z	public: __cde	
00007FFE1D630000	kernel32.dll	系統模組	C:\Windows\System32\kernel32.dll	Unloaded	00007FFE1C5F5820	導出	9	??0?\$_Yarne@Gstd@@QEAA@XZ	public: __cde	
00007FFE1E2A0000	gdi32.dll	系統模組	C:\Windows\System32\gdi32.dll	Unloaded	00007FFE1C618BE0	導出	10	??0?\$_Yarne@_wstd@@QEAA@AEBV01@@Z	public: __cde	
00007FFE1ED90000	ntdll.dll	系統模組	C:\Windows\System32\ntdll.dll	Unloaded	00007FFE1C618C10	導出	11	??0?\$_Yarne@_wstd@@QEAA@PEB_w@Z	public: __cde	

Tools - x64dbg

- 常用功能
 - 呼叫堆疊 (call stack)
 - 記憶體映射 (memory mapping)
 - ...

Tools - x64dbg

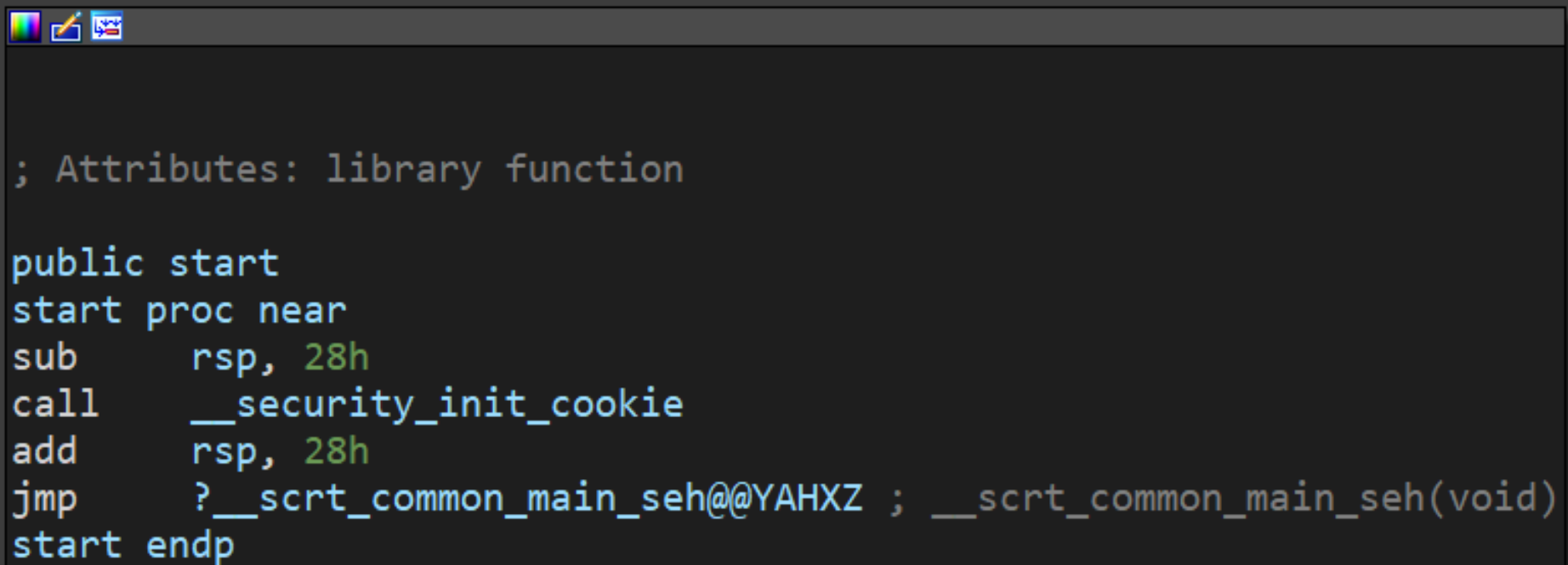
- 外掛
 - ScyllaHide: <https://github.com/x64dbg/ScyllaHide>



Where to start?

main

- Where is main?

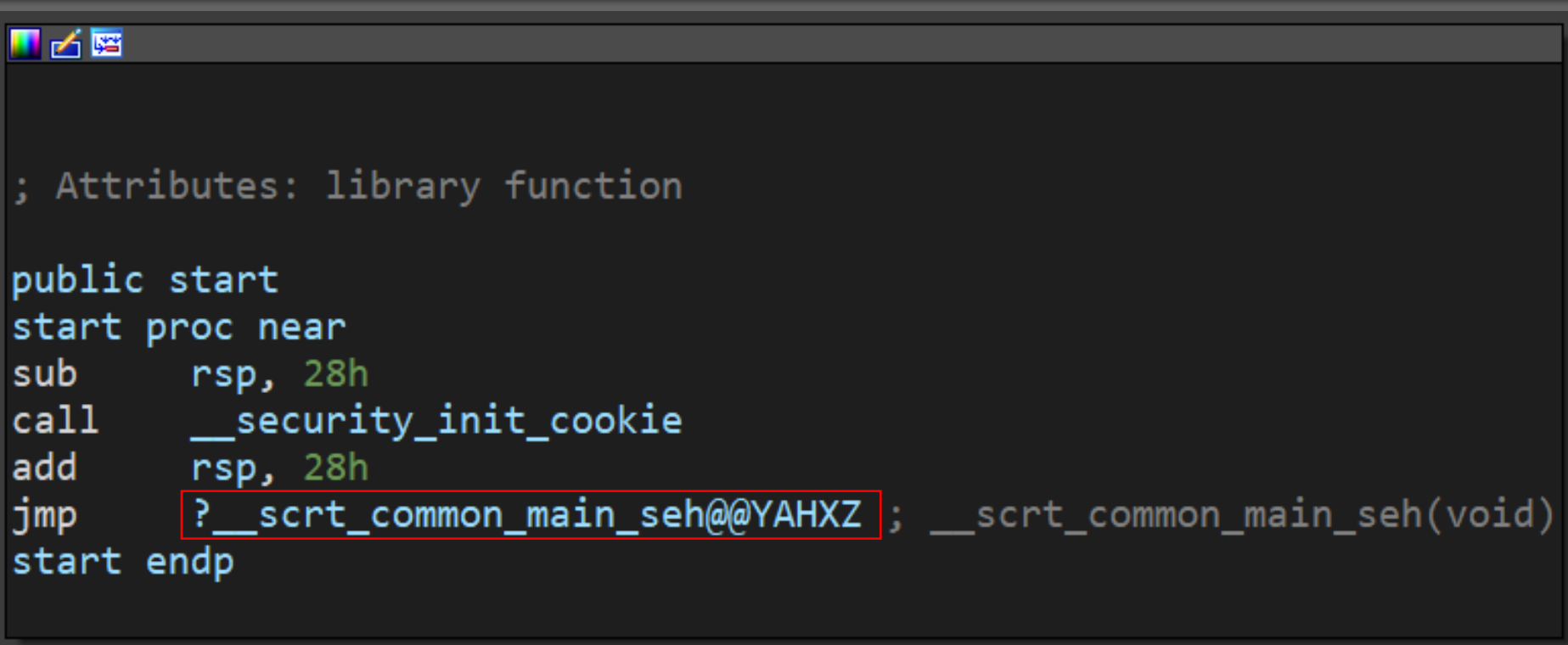


```
; Attributes: library function

public start
start proc near
sub     rsp, 28h
call    __security_init_cookie
add     rsp, 28h
jmp     ?__scrt_common_main_seh@@YAHXZ ; __scrt_common_main_seh(void)
start endp
```


main

- Where is main?
- 在這裡面找有三個參數的 function call



```
; Attributes: library function

public start
start proc near
sub     rsp, 28h
call    __security_init_cookie
add     rsp, 28h
jmp     ?__scrt_common_main_seh@@YAHXZ ; __scrt_common_main_seh(void)
start endp
```

main

```
loc_1400014C2:           ; Microsoft VisualC 64bit universal runtime
call    unknown_libname_11
mov     rdi, rax
call    sub_14000A9E0
mov     rbx, [rax]
call    sub_14000A9D8
mov     r8, rdi           ; envp
mov     rdx, rbx           ; argv
mov     ecx, [rax]        ; argc
call    main
```

Before main

- 比 main 還要早跑的 init 的部分: _initterm_e

```
mov     cs:dword_140024BB0, 1
lea     rdx, unk_1400192C0
lea     rcx, unk_140019288
call    _initterm_e
test    eax, eax
jz      short loc_140001716
```

```
.rdata:00000000140019288 qword_140019288 dq 0 ; DATA XREF: __scr
.rdata:00000000140019290 dq offset sub_1400015C4
.rdata:00000000140019298 dq offset ?post_pgo_initialization@@YAHXZ
.rdata:000000001400192A0 dq offset __acrt_initialize_stdio
.rdata:000000001400192A8 dq offset ?initialize_multibyte@@YAHXZ ; i
.rdata:000000001400192B0 dq offset sub_140010F68
.rdata:000000001400192B8 dq offset __acrt_initialize_fma3
```

Before main

- 比 main 還要早跑的 init 的部分: _initterm

```
loc_140001716:  
lea     rdx, unk_140019280  
lea     rcx, unk_140019268  
call    _initterm  
mov     cs:dword_140024BB0, 2  
jmp     short loc_14000173D
```

```
.rdata:0000000140019268 qword_140019268 dq 0 ; DATA XREF: __scr  
.rdata:0000000140019270 dq offset ?pre_cpp_initialization@@YAXXZ ;  
.rdata:0000000140019278 dq offset sub_1400011D0  
.rdata:0000000140019280 unk_140019280 db 0 ; DATA XREF: __scr
```

After main

- 在 main 結束後才跑的部分, 需用以下 API 註冊要跑的函數
 - atexit
 - _onexit, _onexit_m
 - __dllonexit

Address Space

Address Space

- 目前申請到的記憶體空間
- 從 x64dbg 記憶體映射觀察
- 起始位址 / 大小 / 權限 (E: 可執行; R: 可讀; W: 可寫)

00007FF66C390000	00000000000001000	helloworlds.exe		IMG	-R---	ERWC-
00007FF66C391000	00000000000018000	".text"	可執行代碼	IMG	ER---	ERWC-
00007FF66C3A9000	00000000000008000	".rdata"	唯讀的已初始化的資料	IMG	-R---	ERWC-
00007FF66C3B4000	00000000000002000	".data"	已初始化的資料	IMG	-RW--	ERWC-
00007FF66C3B6000	00000000000002000	".pdata"	例外資料	IMG	-R---	ERWC-
00007FF66C3B8000	00000000000001000	"_RDATA"		IMG	-R---	ERWC-
00007FF66C3B9000	00000000000001000	".rsrc"	資源	IMG	-R---	ERWC-
00007FF66C3BA000	00000000000001000	".reloc"	Base relocations	IMG	-R---	ERWC-

Lab 1

Speed up!



Speed up

- 更快的找到重要程式碼
- 直接通靈程式會做什麼
- 看引用了哪些外部函數來當作通靈依據

Speed up

- socket, connect, recv, send
 - 網路連線
- CreateService, RegCreateKeyEx
 - 猜測在嘗試持久化程式執行
- CryptEncrypt, CryptDecrypt
 - 加解密

Speed up

- VirtualAlloc, VirtualProtect, CreateRemoteThread, CreateProcess, ResumeThread
 - 惡意程式愛用的 API

Speed up

- RegisterClass 家族
 - MFC application 用來註冊 window class 的 API

```
ATOM RegisterClassW(  
    [in] const WNDCLASSW *lpWndClass  
);
```

```
typedef struct tagWNDCLASSA {  
    UINT        style;  
    WNDPROC     lpfnWndProc;  
    int         cbClsExtra;  
    int         cbWndExtra;  
    HINSTANCE   hInstance;  
    HICON        hIcon;  
    HCURSOR     hCursor;  
    HBRUSH       hbrBackground;  
    LPCSTR      lpszMenuName;  
    LPCSTR      lpszClassName;  
} WNDCLASSA, *PWNDCLASSA, *NPWNDCLASSA, *LPWNDCLASSA;
```

Speed up

- RegisterClass 家族
 - MFC application 用來註冊 window class 的 API

Callback function

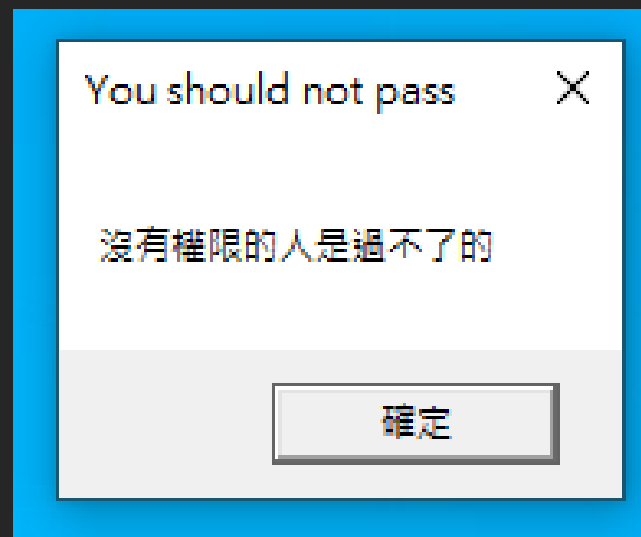
```
ATOM RegisterClassW(  
    [in] const WNDCLASSW *lpWndClass  
);
```

```
typedef struct tagWNDCLASSA {  
    UINT        style;  
    WNDPROC     lpfnWndProc;  
    int         cbClsExtra;  
    int         cbWndExtra;  
    HINSTANCE   hInstance;  
    HICON       hIcon;  
    HCURSOR     hCursor;  
    HBRUSH      hbrBackground;  
    LPCSTR      lpszMenuName;  
    LPCSTR      lpszClassName;  
} WNDCLASSA, *PWNDCLASSA, *NPWNDCLASSA, *LPWNDCLASSA;
```

Speed up

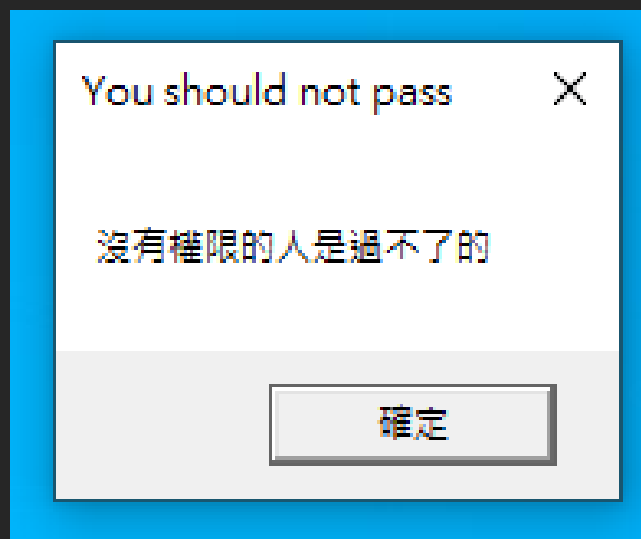
- 在覺得會執行的 API 設定中斷點
- 真的停在中斷點後, 從 call stack 往回找是哪邊呼叫到這個 API

Speed up





Speed up

- 猜是呼叫 MessageBox 家族 API 所創出的 window






Speed up

- 從 IDA Imports 看, 發現有引用

Address	Ordinal	Name 	Library
 000000014000D258		MessageBoxW	USER32

```
.idata:000000014000D258 ; int __stdcall MessageBoxW(HWND hWnd, LP  
.idata:000000014000D258 extrn MessageBoxW:qword
```

 xrefs to MessageBoxW

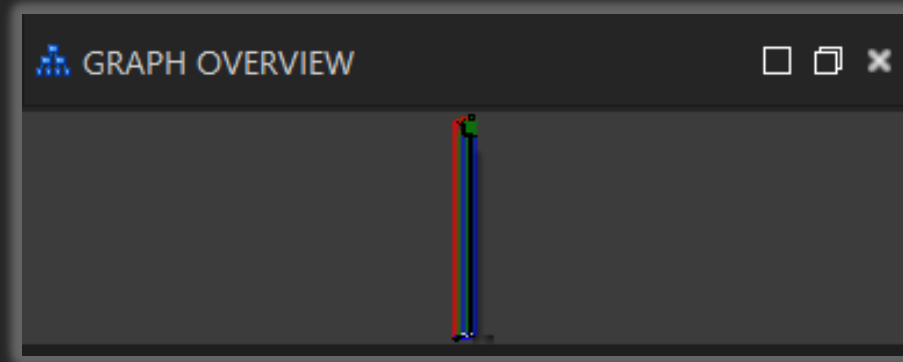
Direction	Type	Address	Text
 Up	p	sub_1400013A0+171E	call cs:MessageBoxW
 Up	r	sub_1400013A0+171E	call cs:MessageBoxW

Speed up

- 往回追到關鍵程式碼區域



```
loc_140002AB8:          ; uType  
xor     r9d, r9d  
mov     rcx, rsi        ; hWnd  
call    cs:MessageBoxW
```



Lab 2

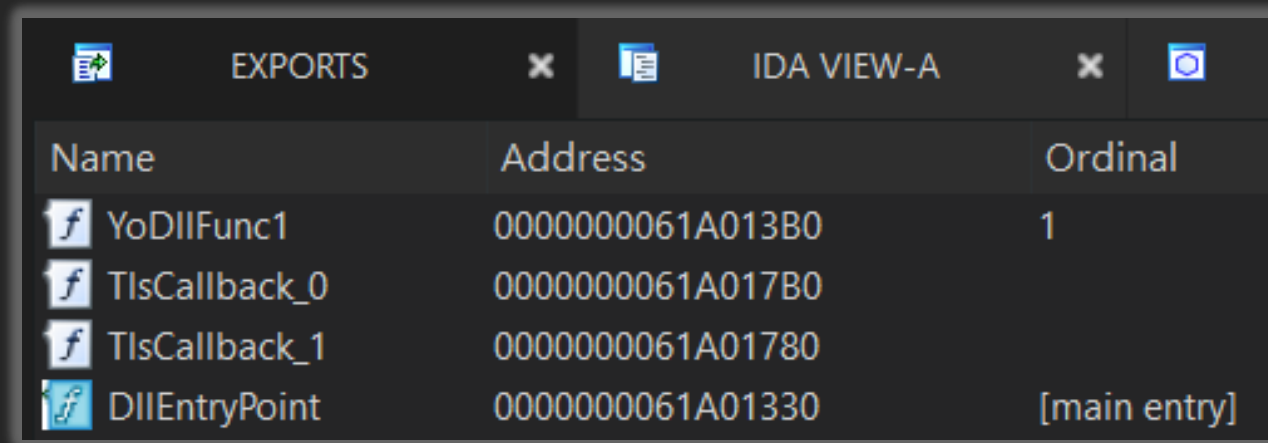
DLL

DLL





- Dynamic-Link Library
- 提供函數給程式使用
- Export function

DLL

- 查看提供了哪些函数



The screenshot shows the 'EXPORTS' window in IDA Pro. The window has a title bar with a file icon, the text 'EXPORTS', a close button, and a maximize button. Below the title bar is a table with three columns: 'Name', 'Address', and 'Ordinal'. The table contains four entries, each with a function icon to its left. The first entry is 'YoDllFunc1' at address '0000000061A013B0' with ordinal '1'. The second is 'TlsCallback_0' at address '0000000061A017B0'. The third is 'TlsCallback_1' at address '0000000061A01780'. The fourth is 'DllEntryPoint' at address '0000000061A01330' with the ordinal '[main entry]'.

Name	Address	Ordinal
 YoDllFunc1	0000000061A013B0	1
 TlsCallback_0	0000000061A017B0	
 TlsCallback_1	0000000061A01780	
 DllEntryPoint	0000000061A01330	[main entry]

DLL

- x64dbg 符號這一區還不錯用
- 看目前引用了哪些 module
- 看 module import/export 哪些函數

CPU		日誌		筆記		中斷點		記憶體映射		呼叫堆疊		SEH鏈		腳本		符號		<> 原始碼			
基址	模組	Party	路徑	位址	類型	序數	符號														
0000000000EC0000	used11.exe	使用者模組	C:\Users\pt\Downloads\d11\used11.exe	0000000000EC93C0	導入		YoD11Func1														
0000000069E20000	otherd11.dll	使用者模組	C:\Users\pt\Downloads\d11\otherD11.dll																		
0000000069E30000	hid11.dll	使用者模組	C:\Users\pt\Downloads\d11\hid11.dll																		
00007FFE1C7E0000	kernelbase.dll	系統模組	C:\Windows\System32\kernelBase.dll																		
00007FFE1D630000	kernel32.dll	系統模組	C:\Windows\System32\kernel32.dll																		
00007FFE1E9B0000	msvcrt.dll	系統模組	C:\Windows\System32\msvcrt.dll																		
00007FFE1ED90000	ntdll.dll	系統模組	C:\Windows\System32\ntdll.dll																		

DLL

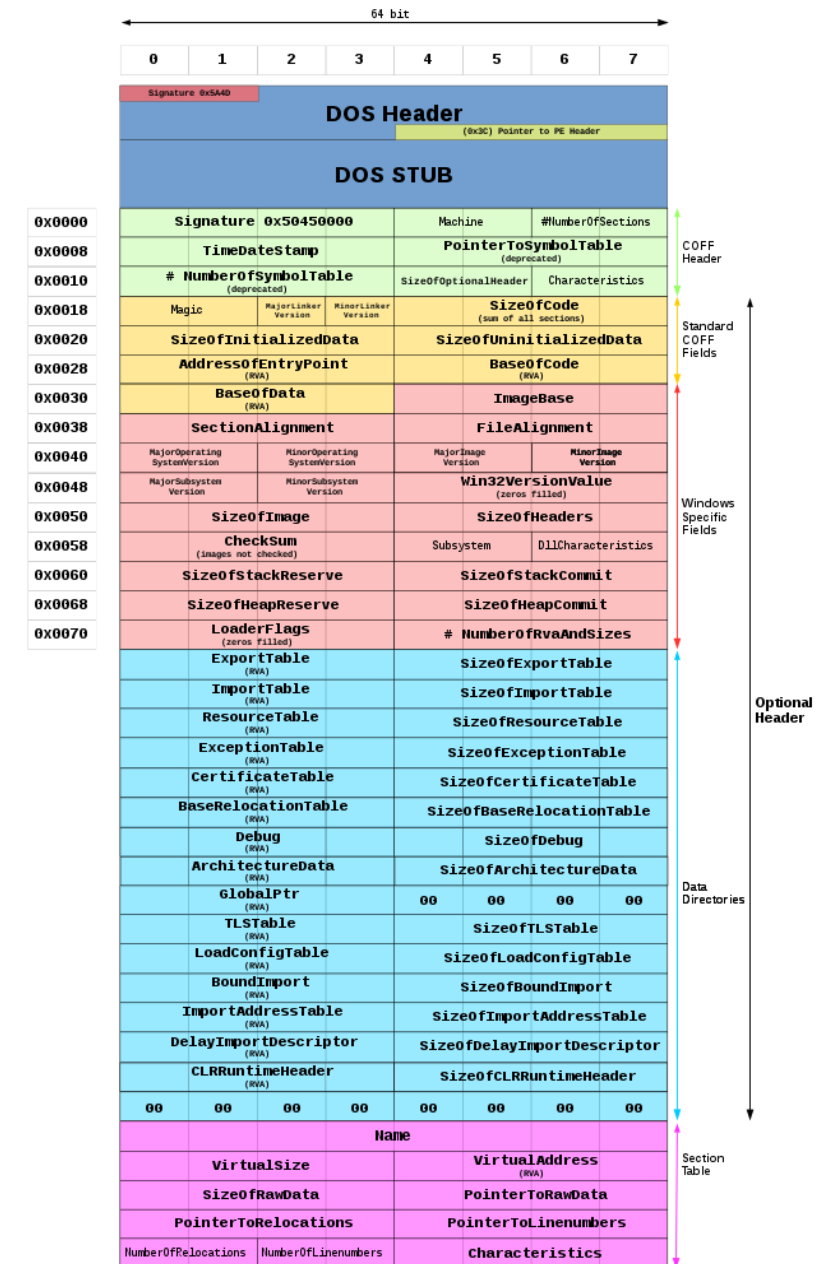
- `DllMain`(`HINSTANCE hinstDLL`, `DWORD fdwReason`, `LPVOID lpReserved`)
- 會在各種時機呼叫到 (e.g. 剛載入時, 要卸載時)
- 可以設定, 也可以不設定

Lab 3

PE format

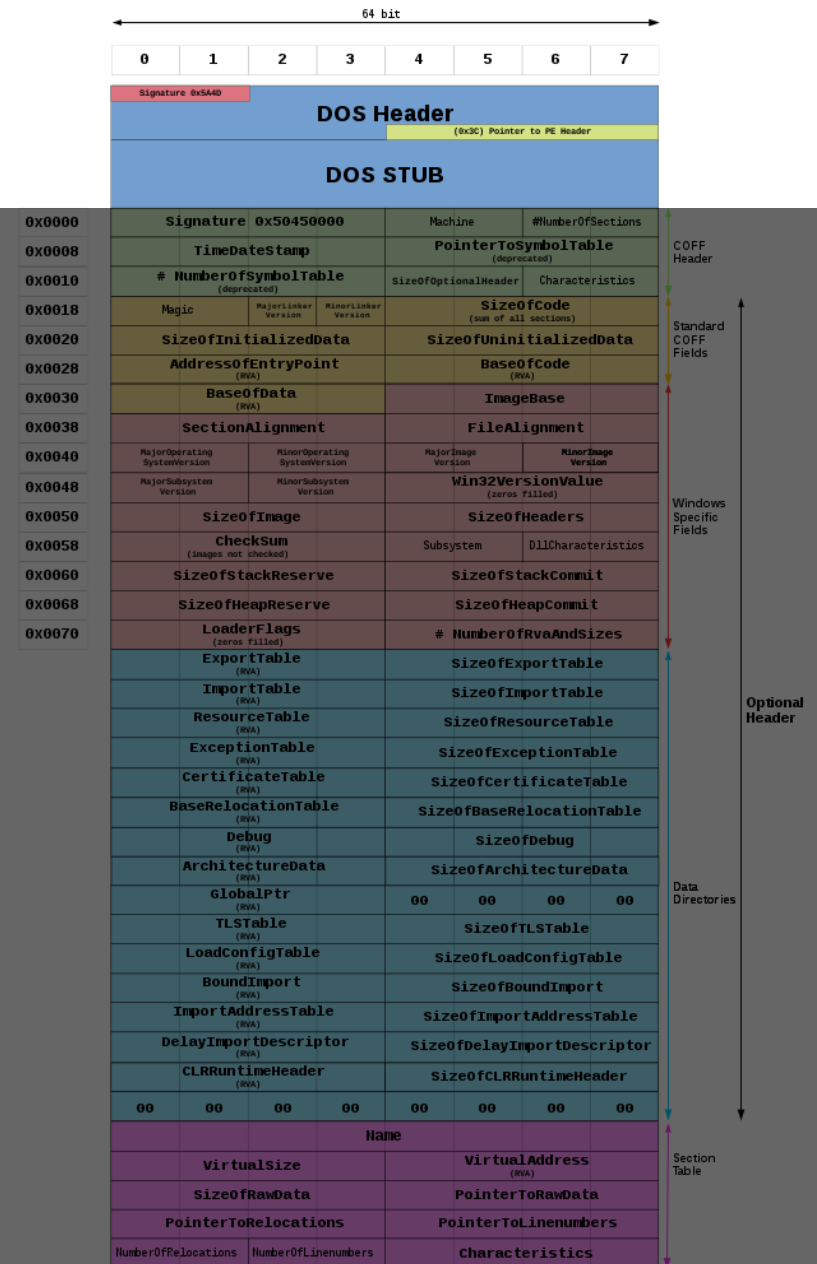
PE format

- 這就是你熟悉的 exe 的結構
- 是不是看了就頭痛
- 讓我們一點一點地拆開來說



PE format

- 首先看最上面的 DOS Header
- 用 PE-Bear 來展示一下



PE format

PE-bear v0.5.4 [C:/Users/pt/Downloads/hello.exe]

File Settings View Compare Info

hello.exe

- DOS Header
- DOS stub
- NT Headers
 - Signature
 - File Header
 - Optional Header
- Section Headers
 - .text
 - EP = 980
 - .rdata
 - .data
 - .pdata
 - .rsrc
 - .reloc

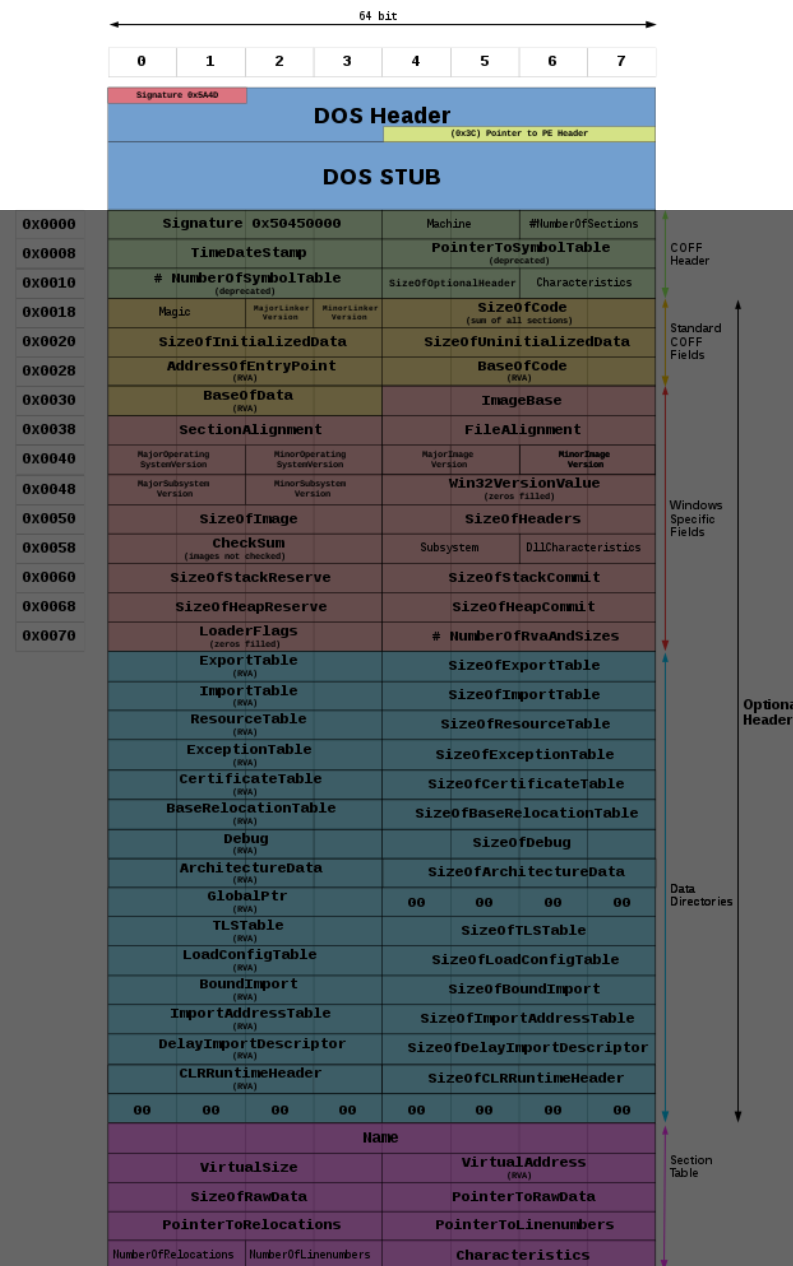
Disasm General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs

Offset	Name	Value
0	Magic number	5A4D
2	Bytes on last page of file	90
4	Pages in file	3
6	Relocations	0
8	Size of header in paragraphs	4
A	Minimum extra paragraphs needed	0
C	Maximum extra paragraphs needed	FFFF
E	Initial (relative) SS value	0
10	Initial SP value	B8
12	Checksum	0
14	Initial IP value	0
16	Initial (relative) CS value	0
18	File address of relocation table	40
1A	Overlay number	0
1C	Reserved words[4]	0, 0, 0, 0
24	OEM identifier (for OEM information)	0
26	OEM information; OEM identifier specific	0
28	Reserved words[10]	0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3C	File address of new exe header	100

Check for updates

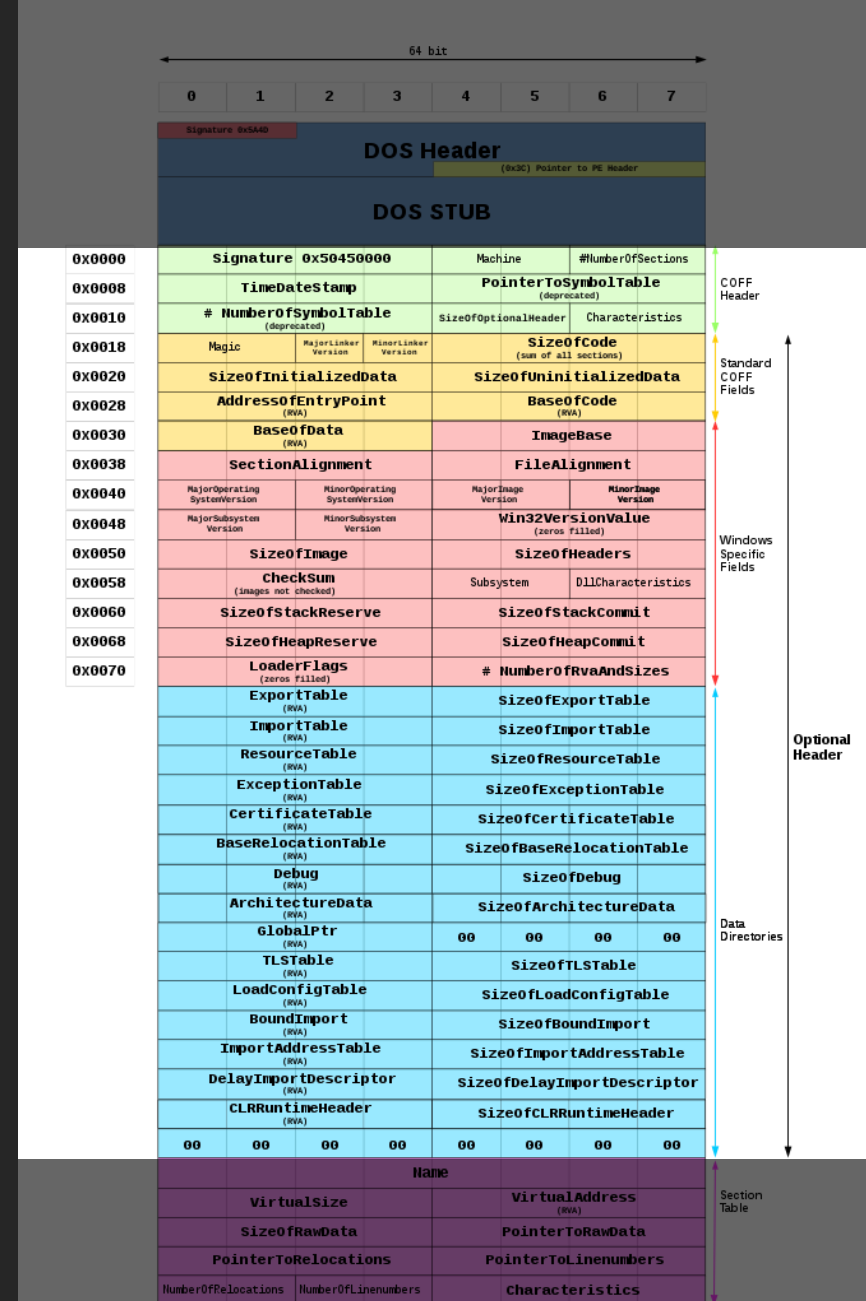
DOS Hdr 以“MZ”作為開頭

紀錄 NT Hdr 偏移量(offset)多少



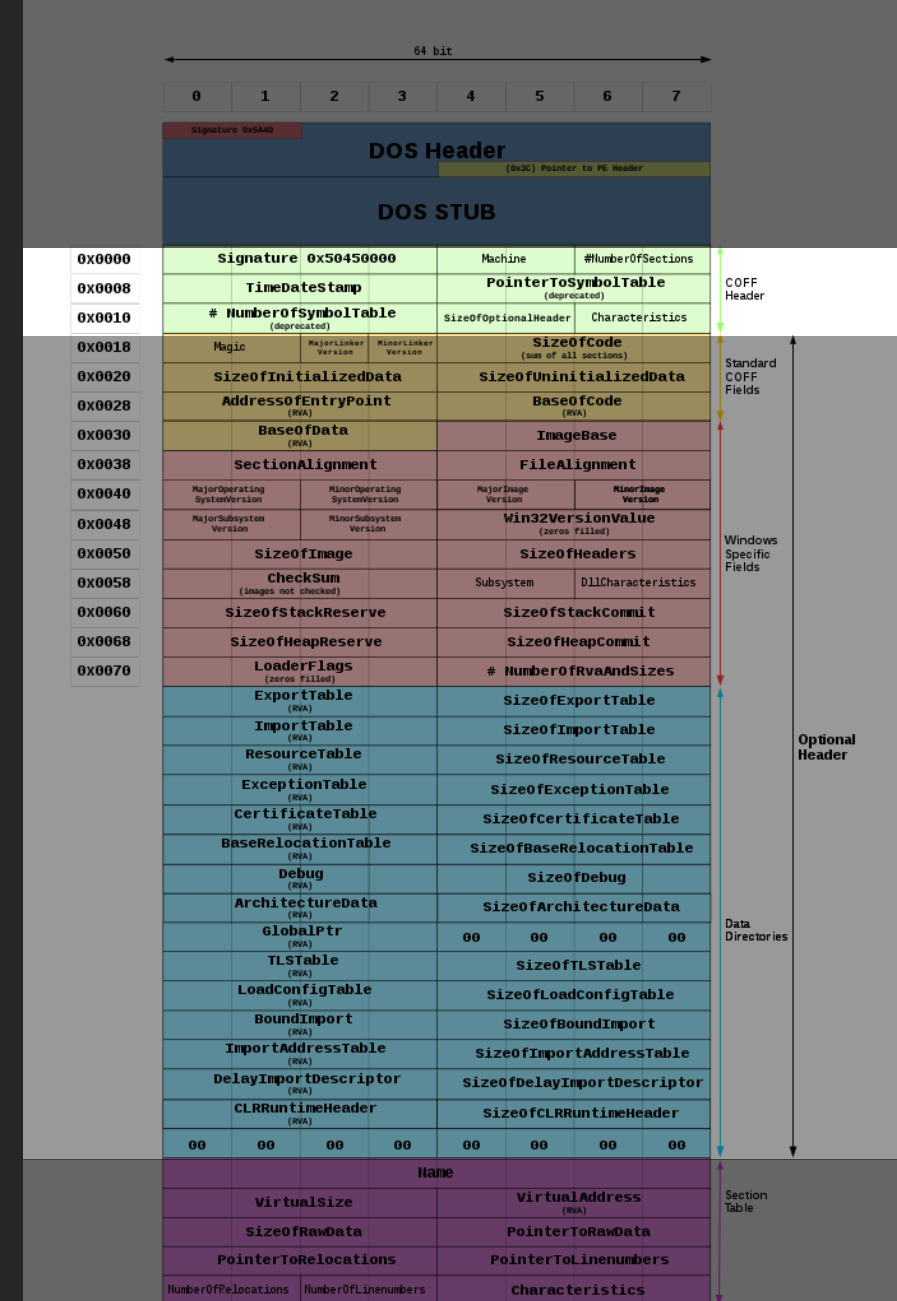
PE format

- 再來是 NT Hdr (或稱 PE Hdr)
- 其包含了



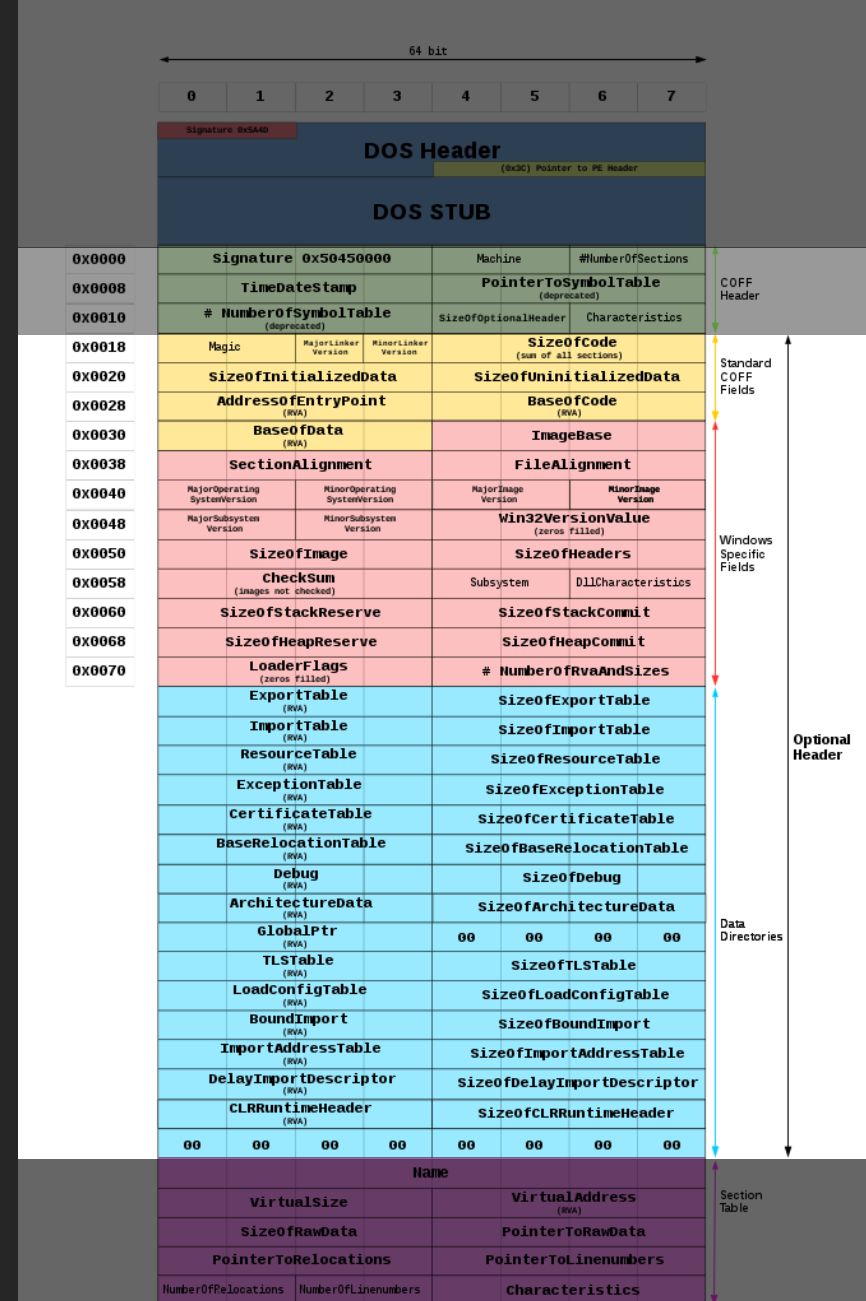
PE format

- 再來是 NT Hdr (或稱 PE Hdr)
- 其包含了
 - COFF Hdr (或稱 File Hdr)



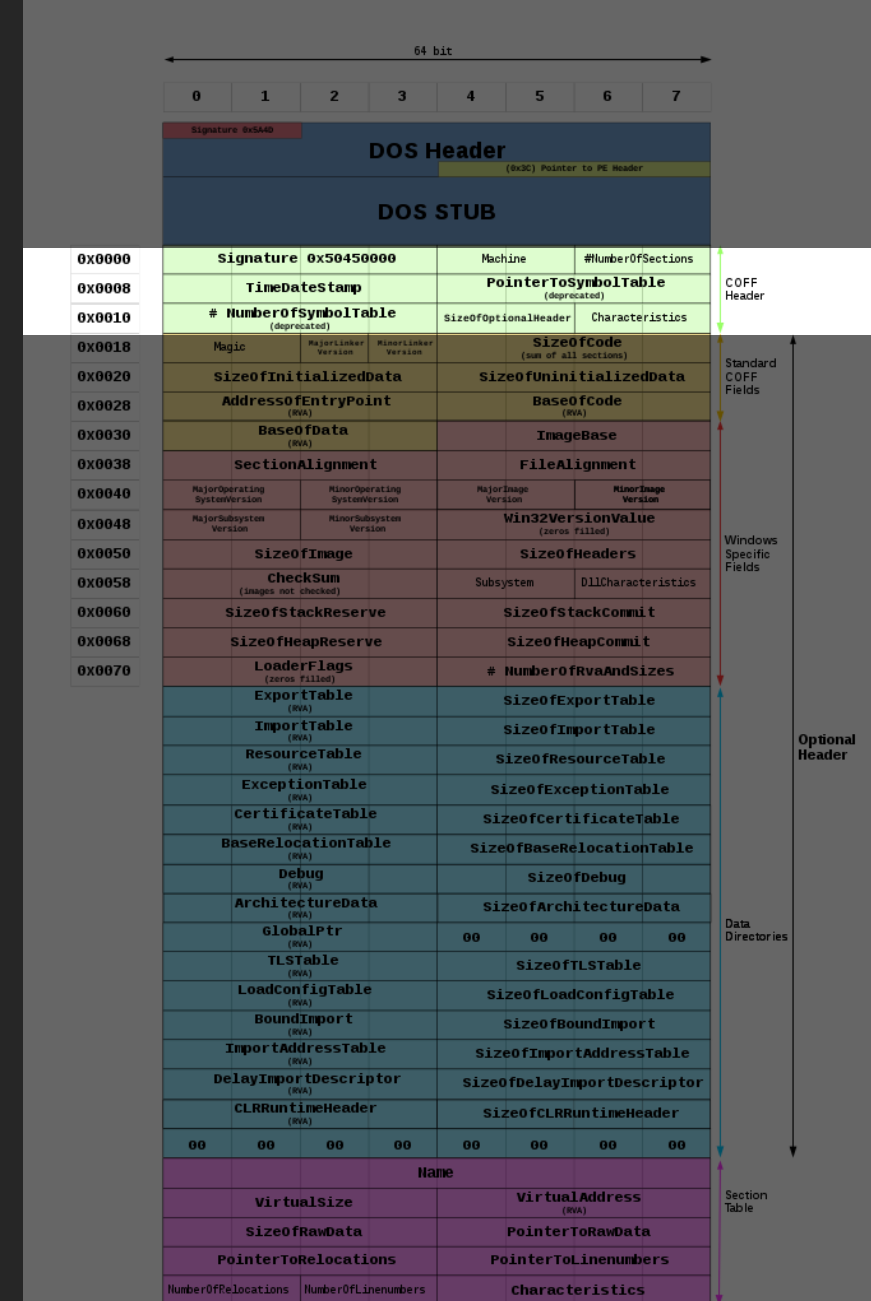
PE format

- 再來是 NT Hdr (或稱 PE Hdr)
- 其包含了
 - COFF Hdr (或稱 File Hdr)
 - Optional Hdr



PE format

- COFF Hdr (或稱 File Hdr)
- 用 PE-Bear 來展示一下



PE format

PE-bear v0.5.4 [C:/Users/pt/Downloads/hello.exe]

File Settings View Compare Info

hello.exe

- DOS Header
- DOS stub
- NT Headers
 - Signature
 - File Header
 - Optional Header
- Section Headers
 - .text
 - EP = 980
 - .rdata
 - .data
 - .pdata
 - .rsrc
 - .reloc

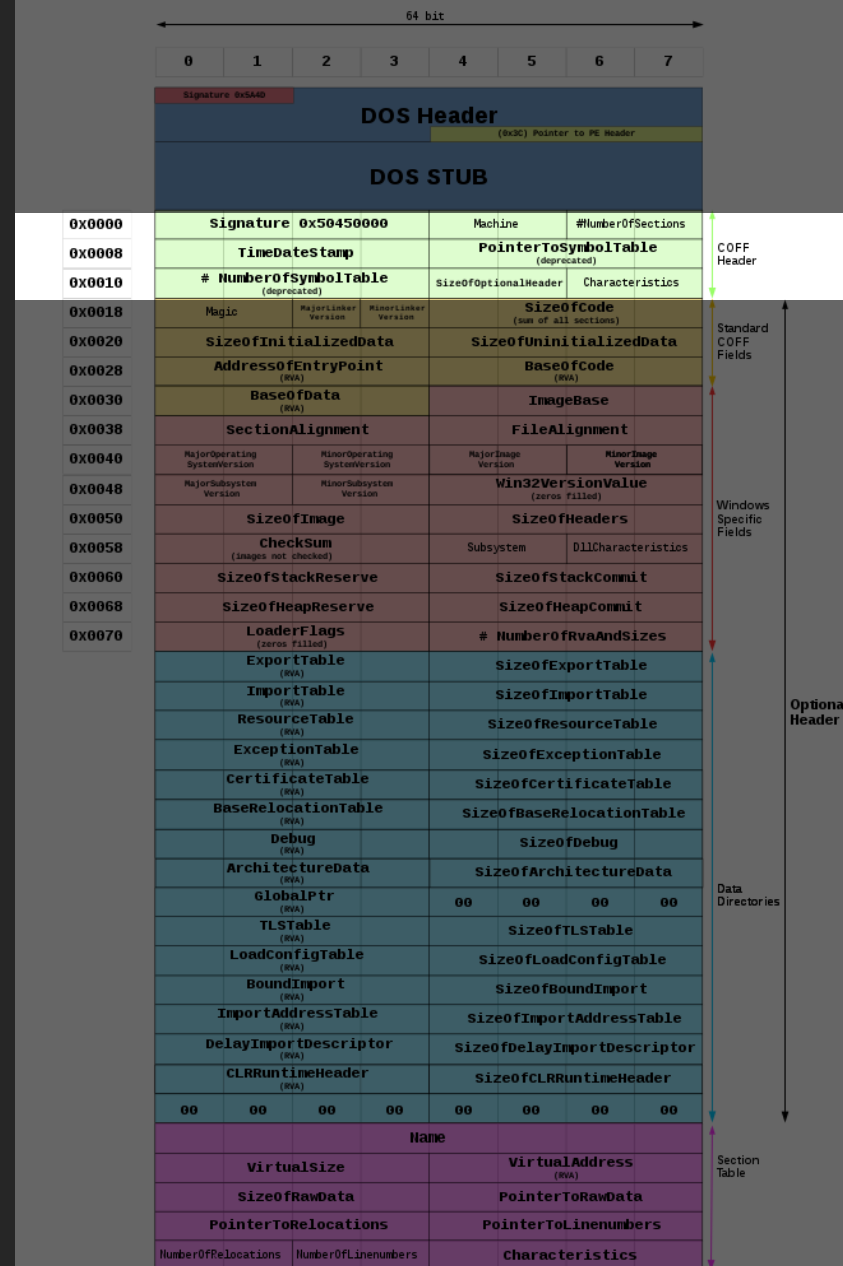
PE Hdr 以“PE”開頭

紀錄有幾個 sections

紀錄 Optional Hdr 多大

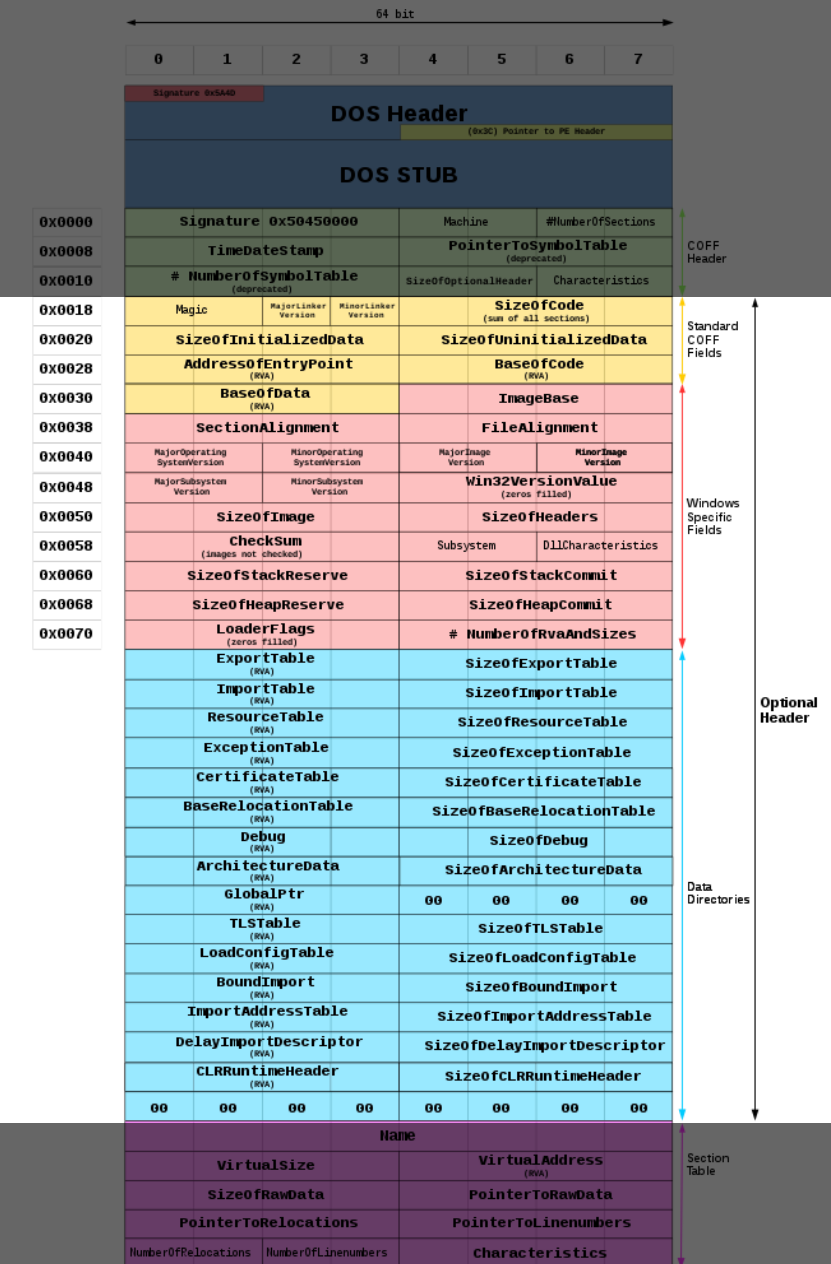
紀錄有哪些特殊設定

Offset	Name	Value	Meaning
104	Machine	8664	AMD64 (K8)
106	Sections Count	6	6
108	Time Date Stamp	6142bfd9	星期四, 16.09.2021 03:54:01 UTC
10C	Ptr to Symbol Table	0	0
110	Num. of Symbols	0	0
114	Size of OptionalHeader	f0	240
116	Characteristics	22	File is executable (i.e. no unresolved external r... App can handle >2gb addresses



PE format

- Optional Hdr
- 用 PE-Bear 來展示一下



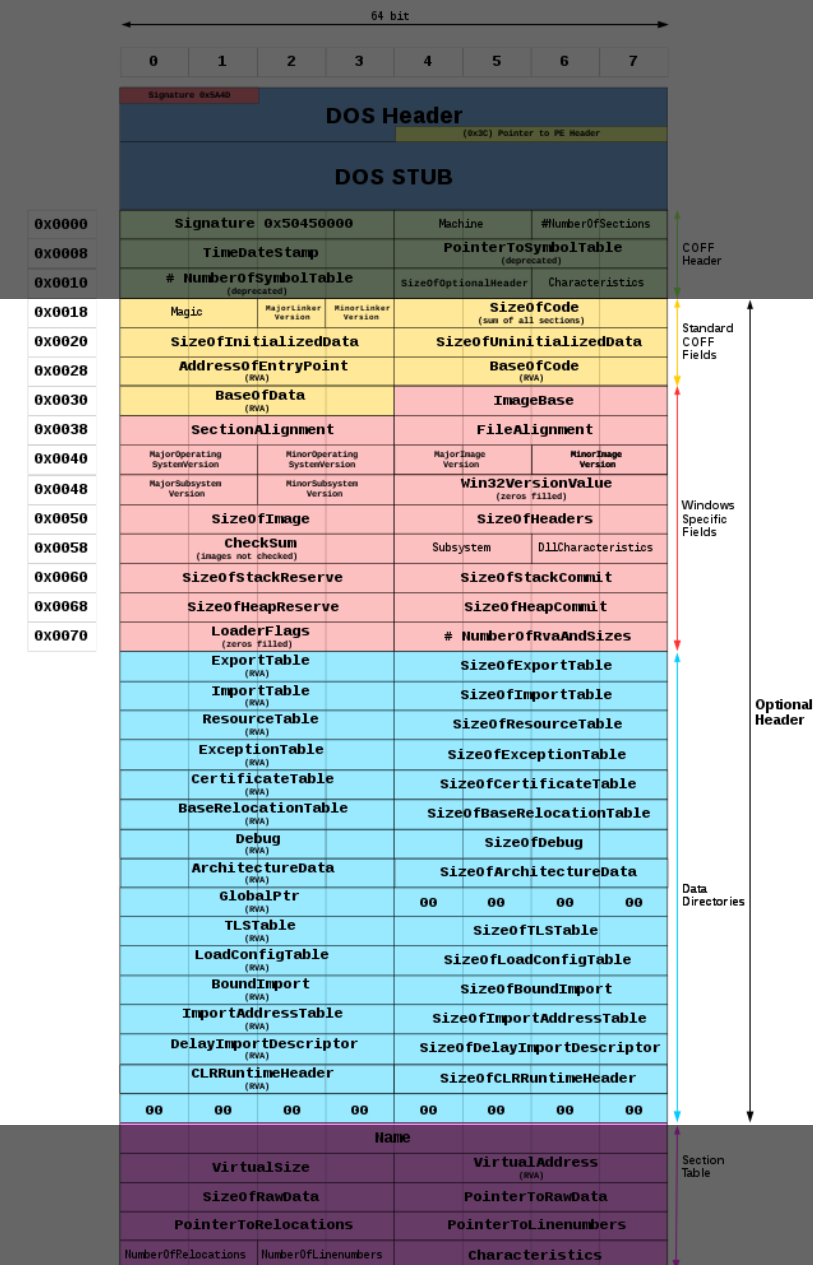
PE format

Disasm	General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs
Offset	Name	Value	Value			
118	Magic	20B	NT64			
11A	Linker Ver. (Major)	E				
11B	Linker Ver. (Minor)	1D				
11C	Size of Code	1200				
120	Size of Initialized Data	2000				
124	Size of Uninitialized Data	0				
128	Entry Point	1580				
12C	Base of Code	1000				
130	Image Base	14000000				
138	Section Alignment	1000				
13C	File Alignment	200				
140	OS Ver. (Major)	6	Windows Vista / Server 2008			
142	OS Ver. (Minor)	0				
144	Image Ver. (Major)	0				
146	Image Ver. (Minor)	0				
148	Subsystem Ver. (Major)	6				
14A	Subsystem Ver. Minor)	0				
14C	Win32 Version Value	0				
150	Size of Image	9000				
154	Size of Headers	400				
158	Checksum	0				
15C	Subsystem	3	Windows console			
15E	DLL Characteristics	8160				
		40	DLL can move			
		100	Image is NX compatible			
		8000	TerminalServer aware			

程式進入點 RVA

程式基址

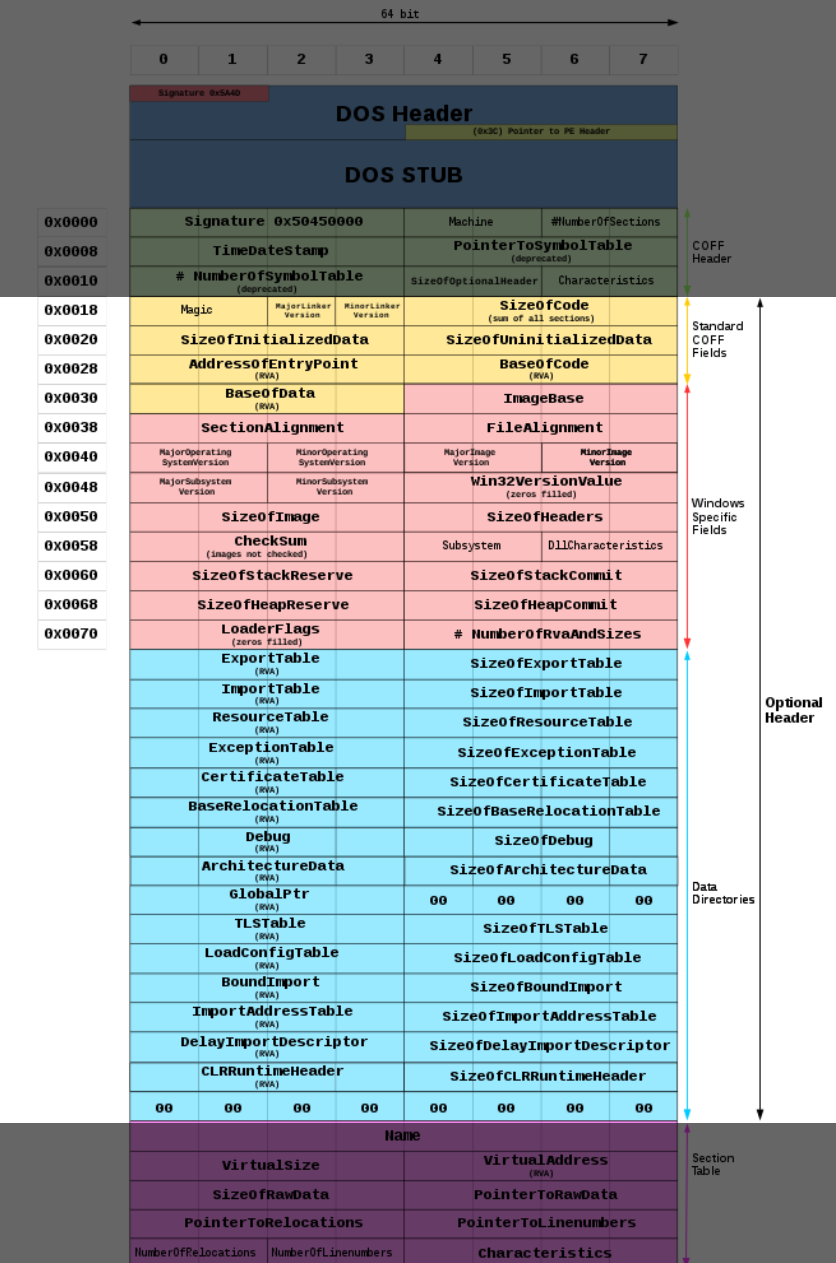
紀錄有哪些特殊設定



PE format

160	Size of Stack Reserve	100000
168	Size of Stack Commit	1000
170	Size of Heap Reserve	100000
178	Size of Heap Commit	1000
180	Loader Flags	0
184	Number of RVAs and Sizes	10
Data Directory		
	Address	Size
188	Export Directory	0
190	Import Directory	396C
198	Resource Directory	7000
1A0	Exception Directory	6000
1A8	Security Directory	0
1B0	Base Relocation Table	8000
1B8	Debug Directory	3310
1C0	Architecture Specific Data	0
1C8	RVA of GlobalPtr	0
1D0	TLS Directory	0
1D8	Load Configuration Directory	3380
1E0	Bound Import Directory in headers	0
1E8	Import Address Table	3000
1F0	Delay Load Import Descriptors	0
1F8	.NET header	0

各種 Directory 位址及大小



PE format

- 解釋一下 RVA (Relative Virtual Address)
- 首先先直接展示一個程式在跑的時候，記憶體位址的樣子

PE format

位址	大小	資訊	內容	類型	保護	初始保護
0000002DAA5A4000	0000000000005C000	Reserved (0000002DA		PRV		-RW--
0000002DAA600000	000000000000FC000	Reserved		PRV		-RW--
0000002DAA6FC000	0000000000004000			PRV	-RW-G	-RW--
000001FAD43A0000	00000000000010000			MAP	-RW--	-RW--
000001FAD43B0000	0000000000001000	\Device\HarddiskVo1		MAP	-R---	-R---
000001FAD43C0000	0000000000001D000			MAP	-R---	-R---
000001FAD43E0000	0000000000004000			MAP	-R---	-R---
000001FAD43F0000	0000000000001000			MAP	-R---	-R---
000001FAD4400000	0000000000002000			PRV	-RW--	-RW--
000001FAD4410000	000000000000C9000	\Device\HarddiskVo1		MAP	-R---	-R---
000001FAD44E0000	0000000000001000			MAP	-R---	-R---
000001FAD44F0000	00000000000010000			PRV	-RW--	-RW--
000001FAD4500000	000000000000F0000	Reserved (000001FAD		PRV		-RW--
000001FAD45F0000	0000000000001000			MAP	-R---	-R---
000001FAD4600000	0000000000001000			MAP	-R---	-R---
00007FF4F0140000	0000000000005000			MAP	-R---	-R---
00007FF4F0145000	000000000000FB000	Reserved (00007FF4F		MAP		-R---
00007FF4F0240000	0000000100020000	Reserved		PRV		-RW--
00007FF5F0260000	0000000002000000	Reserved		PRV		-RW--
00007FF5F2260000	0000000000001000			PRV	-RW--	-RW--
00007FF5F2270000	0000000000001000			MAP	-R---	-R---
00007FF5F2280000	00000000000033000			MAP	-R---	-R---
00007FF676150000	0000000000001000	hello.exe		IMG	-R---	ERWC-
00007FF676151000	0000000000002000	".text"	可執行代碼	IMG	ER---	ERWC-
00007FF676153000	0000000000002000	".rdata"	唯讀的已初始化的資料	IMG	-R---	ERWC-
00007FF676155000	0000000000001000	".data"	已初始化的資料	IMG	-RW--	ERWC-
00007FF676156000	0000000000001000	".pdata"	例外資料	IMG	-R---	ERWC-
00007FF676157000	0000000000001000	".rsrc"	資源	IMG	-R---	ERWC-
00007FF676158000	0000000000001000	".reloc"	Base relocations	IMG	-R---	ERWC-
00007FFB29680000	0000000000001000	vcruntime140.dll		IMG	-R---	ERWC-
00007FFB29681000	00000000000010000	".text"	可執行代碼	IMG	ER---	ERWC-
00007FFB29691000	0000000000004000	".rdata"	唯讀的已初始化的資料	IMG	-R---	ERWC-
00007FFB29695000	0000000000001000	".data"	已初始化的資料	IMG	-RW--	ERWC-
00007FFB29696000	0000000000001000	".pdata"	例外資料	IMG	-R---	ERWC-
00007FFB29697000	0000000000001000	"._RDATA"		IMG	-R---	ERWC-
00007FFB29698000	0000000000001000	".rsrc"	資源	IMG	-R---	ERWC-
00007FFB29699000	0000000000001000	".reloc"	Base relocations	IMG	-R---	ERWC-

PE format

位址	大小	資訊	內容	類型	保護	初始保護
0000002DAA5A4000	0000000000005C000	Reserved (0000002DA		PRV		-RW--
0000002DAA600000	000000000000FC000	Reserved		PRV		-RW--
0000002DAA6FC000	0000000000004000			PRV	-RW-G	-RW--
000001FAD43A0000	00000000000010000			MAP	-RW--	-RW--
000001FAD43B0000	0000000000001000	\Device\HarddiskVo1		MAP	-R---	-R---
000001FAD43C0000	0000000000001D000			MAP	-R---	-R---
000001FAD43E0000	0000000000004000			MAP	-R---	-R---
000001FAD43F0000	0000000000001000			MAP	-R---	-R---
000001FAD4400000	0000000000002000			PRV	-RW--	-RW--
000001FAD4410000	000000000000C9000	\Device\HarddiskVo1		MAP	-R---	-R---
000001FAD44E0000	0000000000001000			MAP	-R---	-R---
000001FAD44F0000	00000000000010000			PRV	-RW--	-RW--
000001FAD4500000	000000000000F0000	Reserved (000001FAD		PRV		-RW--
000001FAD45F0000	0000000000001000			MAP	-R---	-R---
000001FAD4600000	0000000000001000			MAP	-R---	-R---
00007FF4F0140000	0000000000005000			MAP	-R---	-R---
00007FF4F0145000	000000000000FB000	Reserved (00007FF4F		MAP		-R---
00007FF4F0240000	0000000100020000	Reserved		PRV		-RW--
00007FF5F0260000	0000000002000000	Reserved		PRV		-RW--
00007FF5F2260000	0000000000001000			PRV	-RW--	-RW--
00007FF5F2270000	0000000000001000			MAP	-R---	-R---
00007FF5F2280000	0000000000033000			MAP	-R---	-R---
00007FF676150000	0000000000001000	hello.exe		IMG	-R---	ERWC-
00007FF676151000	0000000000002000	".text"	可執行代碼	IMG	ER---	ERWC-
00007FF676153000	0000000000002000	".rdata"	唯讀的已初始化的資料	IMG	-R---	ERWC-
00007FF676155000	0000000000001000	".data"	已初始化的資料	IMG	-RW--	ERWC-
00007FF676156000	0000000000001000	".pdata"	例外資料	IMG	-R---	ERWC-
00007FF676157000	0000000000001000	".rsrc"	資源	IMG	-R---	ERWC-
00007FF676158000	0000000000001000	".reloc"	Base relocations	IMG	-R---	ERWC-
00007FFB29680000	0000000000001000	vcruntime140.dll		IMG	-R---	ERWC-
00007FFB29681000	00000000000010000	".text"	可執行代碼	IMG	ER---	ERWC-
00007FFB29691000	0000000000004000	".rdata"	唯讀的已初始化的資料	IMG	-R---	ERWC-
00007FFB29695000	0000000000001000	".data"	已初始化的資料	IMG	-RW--	ERWC-
00007FFB29696000	0000000000001000	".pdata"	例外資料	IMG	-R---	ERWC-
00007FFB29697000	0000000000001000	"._RDATA"		IMG	-R---	ERWC-
00007FFB29698000	0000000000001000	".rsrc"	資源	IMG	-R---	ERWC-
00007FFB29699000	0000000000001000	".reloc"	Base relocations	IMG	-R---	ERWC-

VA RVA

- 解釋 RVA (Relative Virtual Address) 之前
- 先解釋什麼是 VA (Virtual Address)
- 做一下小實驗，如果執行兩個 hello.exe，記憶體位址分布長怎樣？

VA RVA

- 兩個 process 的記憶體位址有重疊耶?!
- 如果改掉 A process 記憶體內容 (地址重疊的部分), B process 的內容也會被改嗎?
- 實驗一下, 答案是不會的
- 所以那個記憶體位址到底是啥

VA RVA

- 其實我們的程式所看到的記憶體位址，都是假的
- 都是虛擬記憶體位址 (Virtual Address)



VA RVA

- 那為什麼要這麼複雜?
- 如果程式都能直接碰到實體記憶體位址 PA (Physical Address)
- 你要怎麼知道這個 PA 有沒有被其他程式占用?
- 這個問題很難，但現代 OS 幫你搞定了這個問題
- OS 只給你 VA，實際上存取時，OS 有他的方式，能夠 VA <-> PA

VA RVA

- 正常狀況下 A process 的 0x55665566 VA
- 跟 B process 的 0x55665566 VA
- 不是對應到同一個 PA
- 解釋了剛剛的實驗結果

VA RVA

- 搞懂 VA 了，可以講 RVA 了
- 只是一個方便 PE 結構不用寫這麼多字的東西
- $VA = ImageBase + RVA$
- 第一條指令位址 $VA = ImageBase + \text{Entry point RVA}$

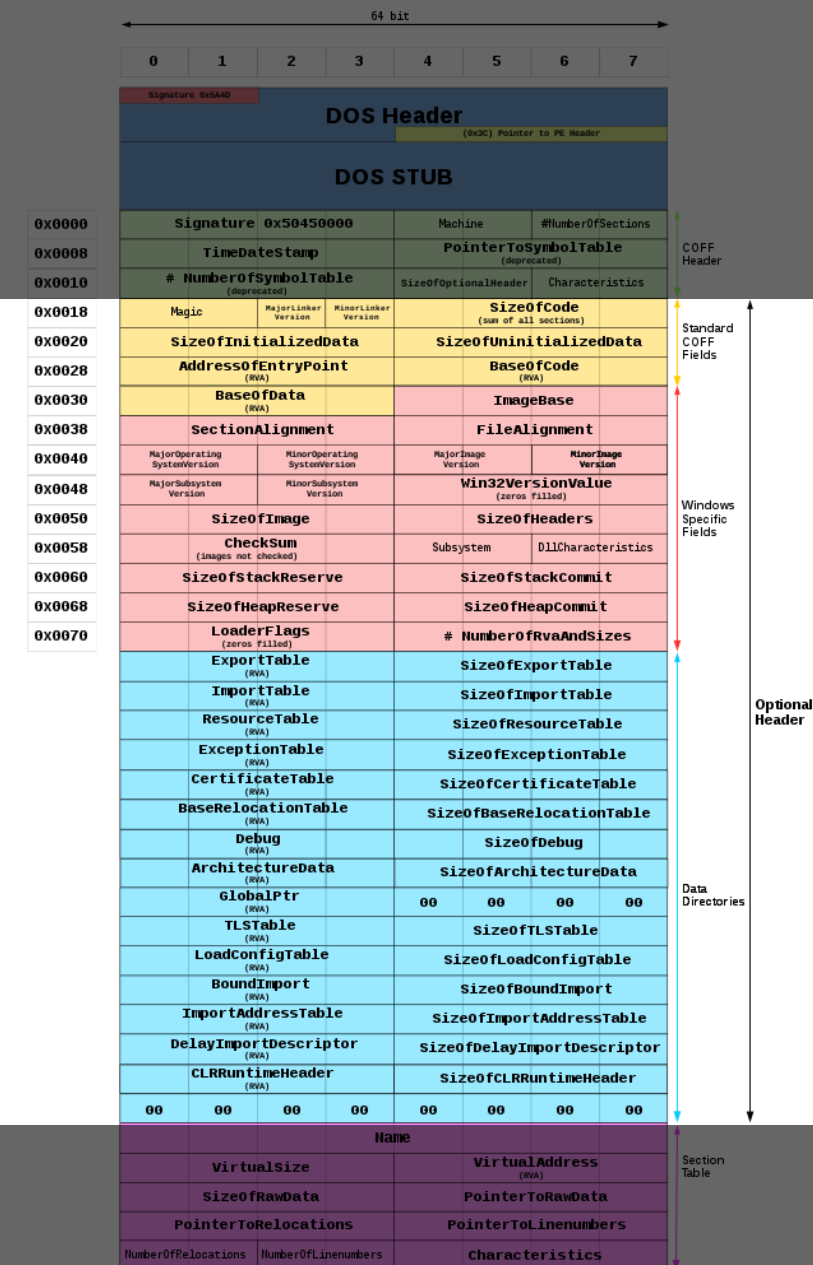
VA RVA

Disasm	General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs
Offset	Name	Value	Value			
118	Magic	20B	NT64			
11A	Linker Ver. (Major)	E				
11B	Linker Ver. (Minor)	1D				
11C	Size of Code	1200				
120	Size of Initialized Data	2000				
124	Size of Uninitialized Data	0				
128	Entry Point	1580				
12C	Base of Code	1000				
130	Image Base	14000000				
138	Section Alignment	1000				
13C	File Alignment	200				
140	OS Ver					
142	OS Ver					
144	Image Ver. (Major)	0				
146	Image Ver. (Minor)	0				
148	Subsystem Ver. (Major)	6				
14A	Subsystem Ver. Minor)	0				
14C	Win32 Version Value	0				
150	Size of Image	9000				
154	Size of Headers	400				
158	Checksum	0				
15C	Subsystem	3	Windows console			
15E	DLL Characteristics	8160				
		40	DLL can move			
		100	Image is NX compatible			
		8000	TerminalServer aware			

程式進入點 RVA

程式基址

第一條指令位址 = 0x140001580



ASLR

- 可是實驗中, 我們的第一條指令位址顯然不是剛算的
- 原因是 ASLR (Address Space Layout Randomization)
- 記憶體位址每次執行時都是固定的話, 會有安全問題
- 啟用了 ASLR, OS 就會隨機產生 ImageBase, 原本提供的 ImageBase 就被忽略了 QQ

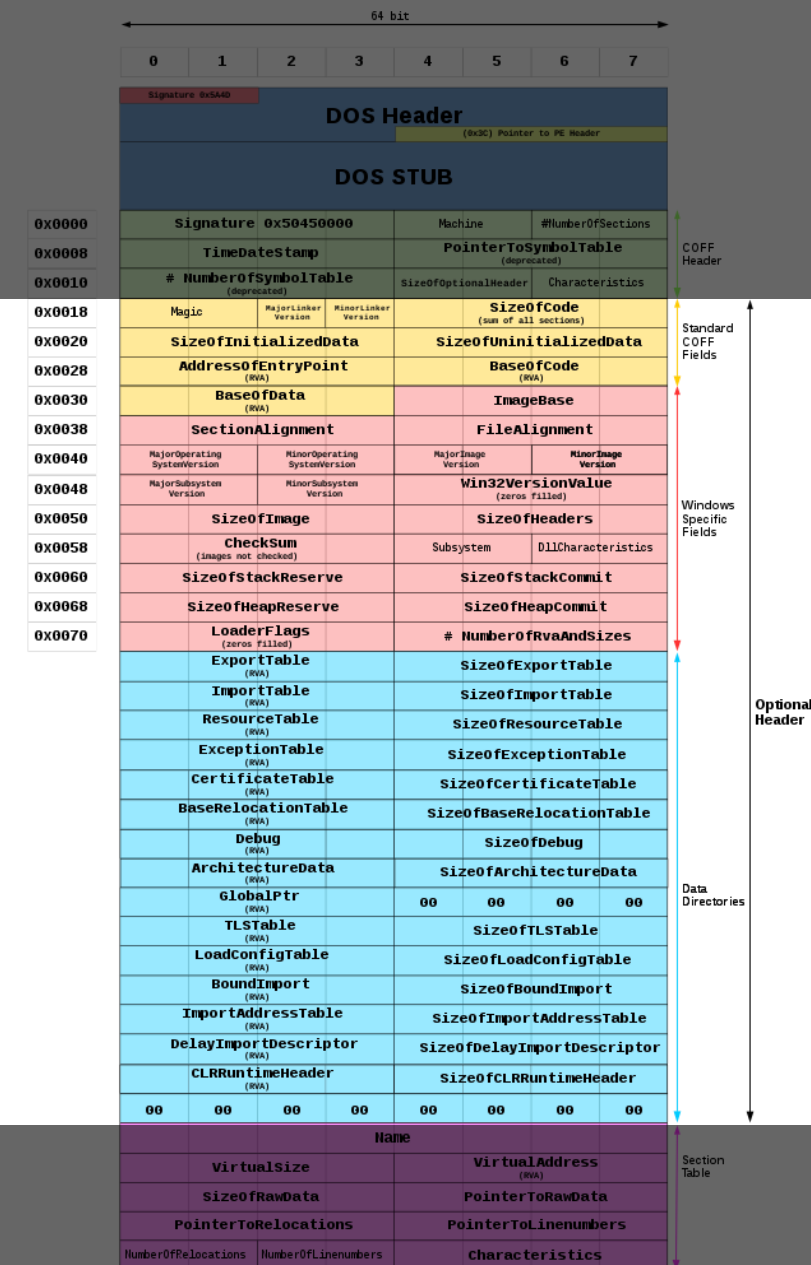
ASLR

- 第一條指令位址 $VA = \text{ASLR 隨機產生的 ImageBase} + \text{進入點 RVA}$
- 那麼要怎麼知道 ASLR 有沒有開啟呢?

ASLR

Disasm	General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs	Im
Offset	Name	Value	Value				
118	Magic	20B	NT64				
11A	Linker Ver. (Major)	E					
11B	Linker Ver. (Minor)	1D					
11C	Size of Code	1200					
120	Size of Initialized Data	2000					
124	Size of Uninitialized Data	0					
128	Entry Point	1580					
12C	Base of Code	1000					
130	Image Base	140000000					
138	Section Alignment	1000					
13C	File Alignment	200					
140	OS Ver. (Major)	6	Windows Vista / Server 2008				
142	OS Ver. (Minor)	0					
144	Image Ver. (Major)	0					
146	Image Ver. (Minor)	0					
148	Subsystem Ver. (Major)	6					
14A	Subsystem Ver. Minor)	0					
14C	Win32 Version Value	0					
150	Size of Image	9000					
154	Size of Headers	400					
158	Checksum	0					
15C	Subsystem	3	Windows console				
15E	DLL Characteristics	8160					
		40	DLL can move				
		100	Image is NX compatible				
		8000	TerminalServer aware				

若有 DLL can move 就是有啟用 ASLR



ASLR

- 如果把程式裡的這個 bit 拔掉, 就可以關掉 ASLR 了
- 來實驗一下!

ASLR

- DLL can move 是 0x40
- 減去 0x40 就是把它拔掉

PE-bear v0.5.4 [C:\Users\pt\Desktop\hello.exe]

File Settings View Compare Info

hello.exe*

- DOS Header
- DOS stub
- NT Headers
 - Signature
 - File Header
 - Optional Header
- Section Headers
- Sections
 - .text
 - EP = 980
 - .rdata
 - .data
 - .pdata
 - .rsrc
 - .reloc

Disasm General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Import

Offset	Name	Value	Value
15E	DLL Characteristics	8160	
		40	DLL can move
		100	Image is NX compatible
		8000	TerminalServer aware

Disasm General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Import

Offset	Name	Value	Value
15E	DLL Characteristics	8120	
		100	Image is NX compatible
		8000	TerminalServer aware

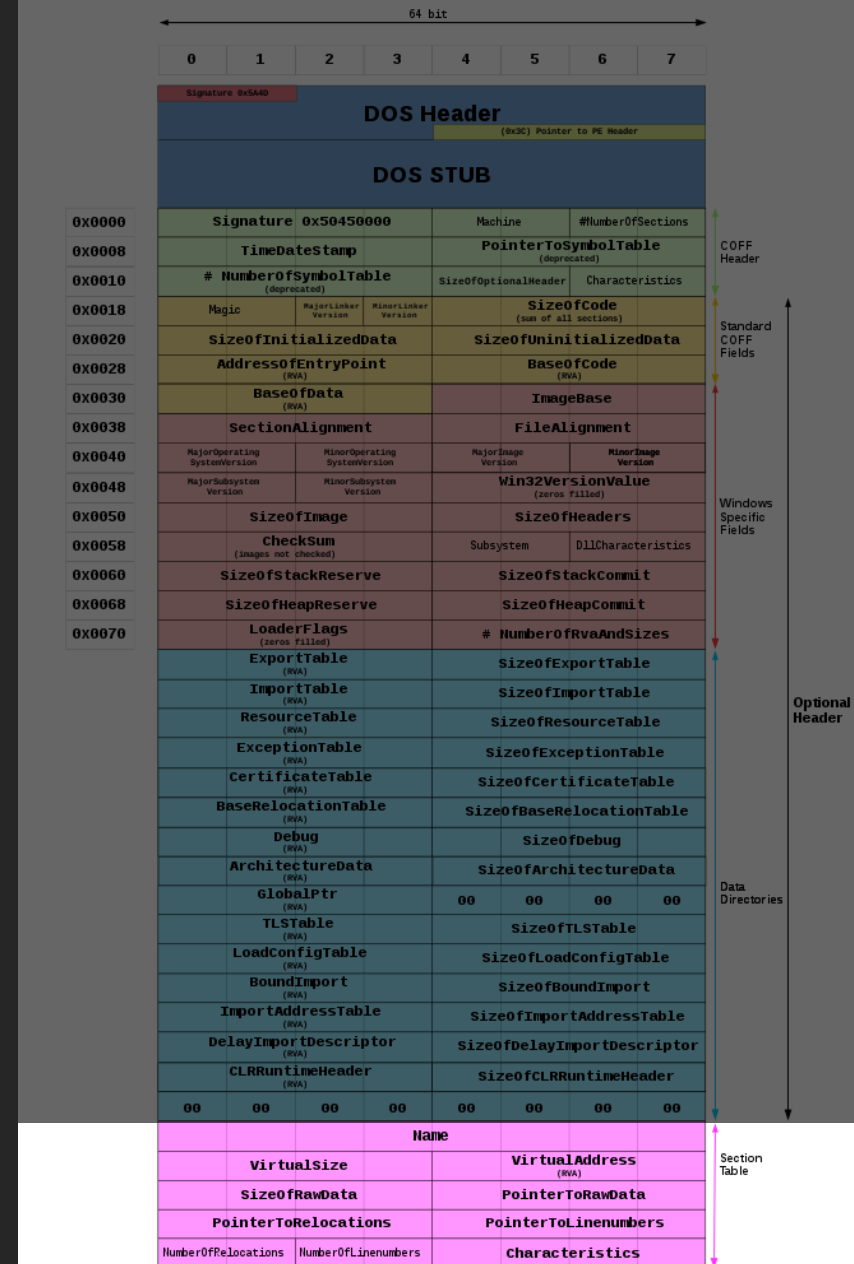
ASLR

- 記得另存一個新程式
- 再度執行看看

0000000140000000	00000000000001000	hello_noaslr.exe
0000000140001000	00000000000002000	".text"
0000000140003000	00000000000002000	".rdata"
0000000140005000	00000000000001000	".data"
0000000140006000	00000000000001000	".pdata"
0000000140007000	00000000000001000	".rsrc"
0000000140008000	00000000000001000	".reloc"

PE format

- Section Hdr
- 在右邊的圖是對應 Section Table
- 其實會有多個 Section Hdr
- 用 PE-Bear 來展示一下



PE format

PE-bear v0.5.4 [C:/Users/pt/Desktop/hello.exe]

File Settings View Compare Info

hello.exe

DOS Header

DOS stub

NT Headers

Signature

File Header

Optional Header

Section Headers

Sections

.text

EP = 980

.rdata

.data

.pdata

.rsrc

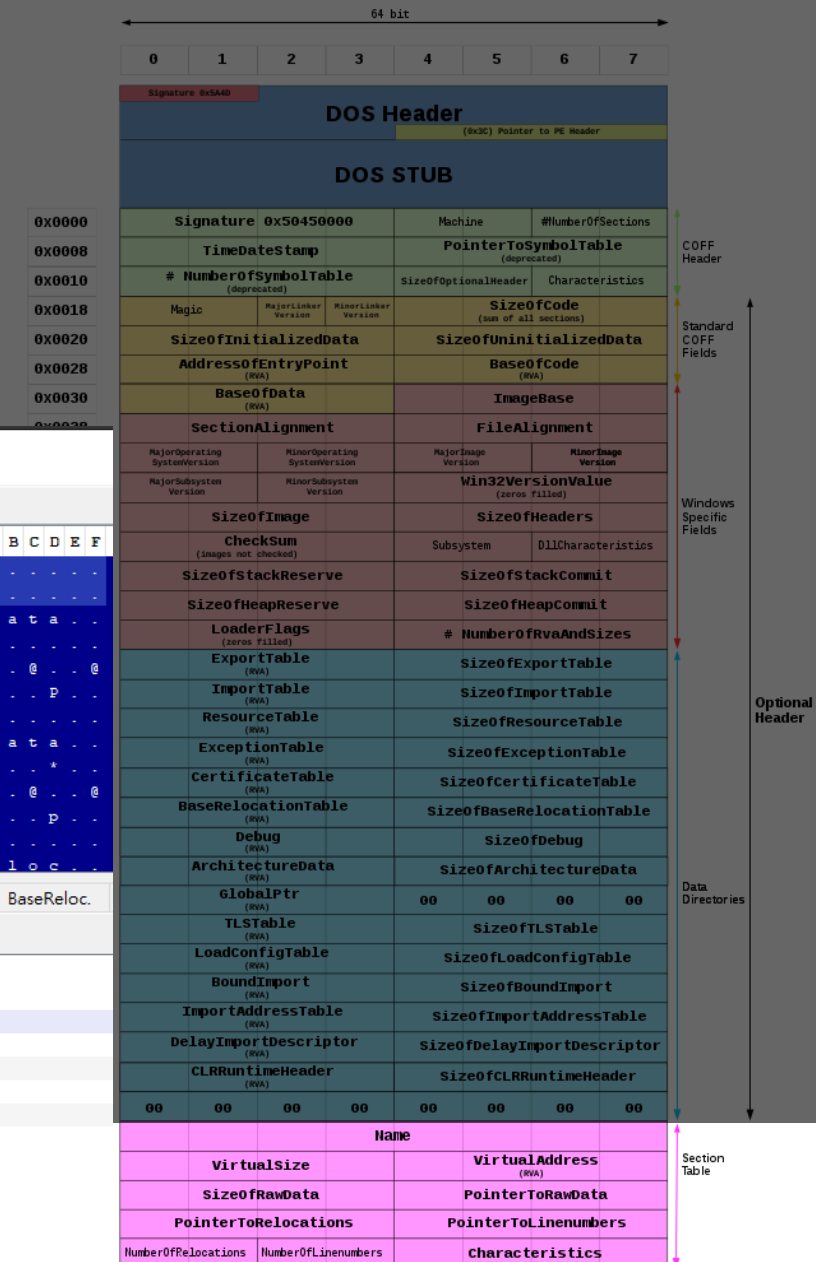
.reloc

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
208	2E	74	65	78	74	00	00	00	4C	10	00	00	00	10	00	00
218	00	12	00	00	00	04	00	00	00	00	00	00	00	00	00	00
228	00	00	00	00	20	00	00	60	2E	72	64	61	74	61	00	00
238	40	11	00	00	00	30	00	00	00	12	00	00	00	16	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	40
258	2E	64	61	74	61	00	00	00	48	06	00	00	00	50	00	00
268	00	02	00	00	00	28	00	00	00	00	00	00	00	00	00	00
278	00	00	00	00	40	00	00	C0	2E	70	64	61	74	61	00	00
288	98	01	00	00	00	60	00	00	00	02	00	00	00	2A	00	00
298	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	40
2A8	2E	72	73	72	63	00	00	00	E0	01	00	00	00	70	00	00
2B8	00	02	00	00	00	2C	00	00	00	00	00	00	00	00	00	00
2C8	00	00	00	00	40	00	00	40	2E	72	65	6C	6F	63	00	00

Disasm: Headers to [.text] General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Imports Resources Exception BaseReloc.

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
▼ .text	400	1200	1000	104C	60000020	0	0	0
> 1600	^	204C	^	r-x				
> .rdata	1600	1200	3000	1140	40000040	0	0	0
> .data	2800	200	5000	648	C0000040	0	0	0
> .pdata	2A00	200	6000	198	40000040	0	0	0
> .rsrc	2C00	200	7000	1E0	40000040	0	0	0
> .reloc	2E00	200	8000	2C	42000040	0	0	0

.text section hdr



PE format

PE-bear v0.5.4 [C:/Users/pt/Desktop/hello.exe]

File Settings View Compare Info

hello.exe

DOS Header
 DOS stub
 NT Headers
 Signature
 File Header
 Optional Header
 Section Headers

Sections
 .text
 EP = 980
 .rdata
 .data
 .pdata
 .rsrc
 .reloc

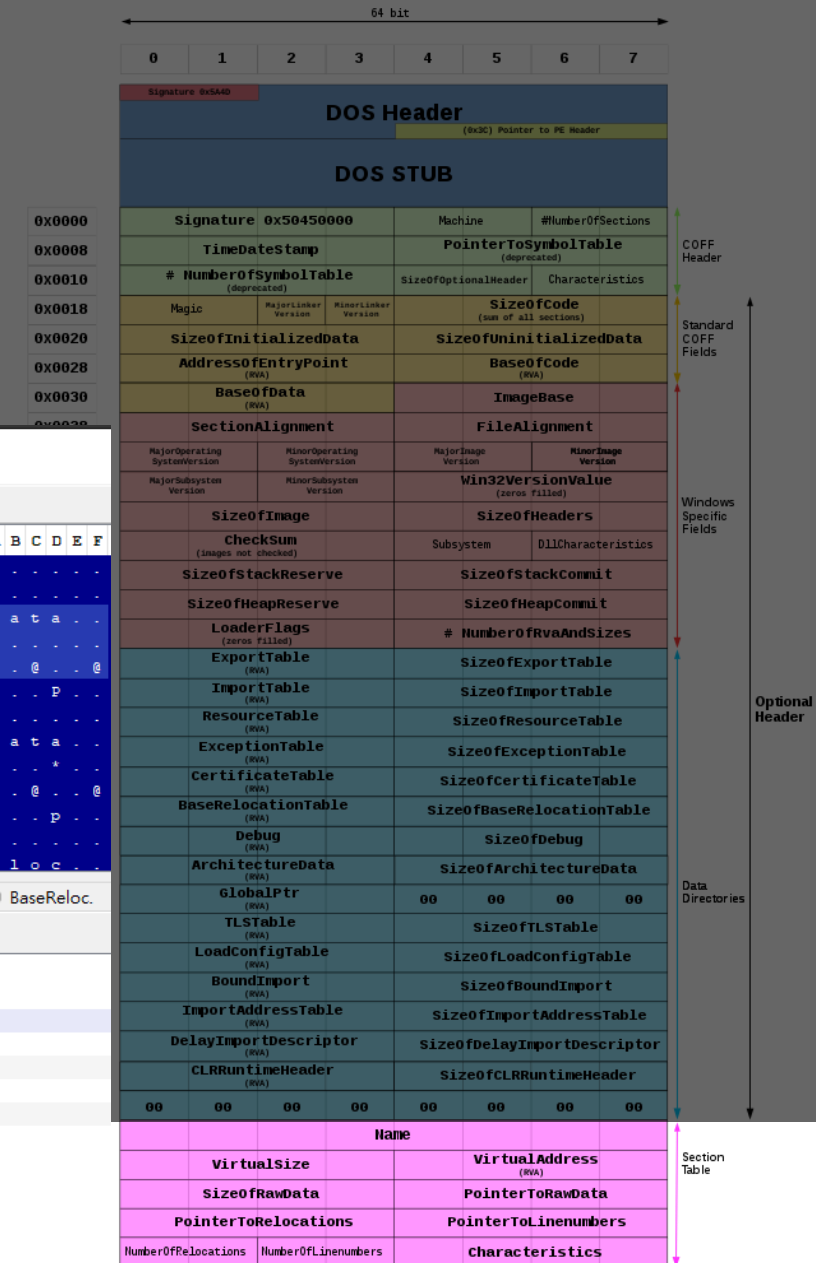
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
208	2E	74	65	78	74	00	00	00	00	4C	10	00	00	00	10	00	00																
218	00	12	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00																
228	00	00	00	00	20	00	00	60	2E	72	64	61	74	61	00	00																	
238	40	11	00	00	00	30	00	00	00	12	00	00	00	16	00	00																	
248	00	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	40																
258	2E	64	61	74	61	00	00	00	48	06	00	00	00	50	00	00																	
268	00	02	00	00	00	28	00	00	00	00	00	00	00	00	00	00																	
278	00	00	00	00	40	00	00	C0	2E	70	64	61	74	61	00	00																	
288	98	01	00	00	00	60	00	00	00	02	00	00	00	00	2A	00	00																
298	00	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	40																
2A8	2E	72	73	72	63	00	00	00	E0	01	00	00	00	70	00	00																	
2B8	00	02	00	00	00	2C	00	00	00	00	00	00	00	00	00	00																	
2C8	00	00	00	00	40	00	00	40	2E	72	65	6C	6F	63	00	00																	

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	.	t	e	x	t	L

Disasm: Headers to [.text] General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Imports Resources Exception BaseReloc.

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
▼ .text	400	1200	1000	104C	60000020	0	0	0
> .rdata	1600	1200	204C	1140	r-x	0	0	0
> .data	2800	200	5000	648	C0000040	0	0	0
> .pdata	2A00	200	6000	198	40000040	0	0	0
> .rsrc	2C00	200	7000	1E0	40000040	0	0	0
> .reloc	2E00	200	8000	2C	42000040	0	0	0

.rdata section hdr



PE format

- 解釋一下 Section Hdr
- 程式檔案的 Raw Addr 開始的 Raw size 個 Bytes 會映射到 Virtual Addr 開始的 Virtual Size 個 Bytes
 - Virtual Addr 是 RVA
- Characteristics 設定了該區記憶體位址的權限

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
▼ .text	400	1200	1000	104C	60000020	0	0	0
> 1600	^	204C	^	r-x				
> .rdata	1600	1200	3000	1140	40000040	0	0	0
> .data	2800	200	5000	648	C0000040	0	0	0
> .pdata	2A00	200	6000	198	40000040	0	0	0
> .rsrc	2C00	200	7000	1E0	40000040	0	0	0
> .reloc	2E00	200	8000	2C	42000040	0	0	0

PE format

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
▼ .text	400	1200	1000	104C	60000020	0	0	0
> .rdata	1600	1200	3000	1140	40000040	0	0	0
> .data	2800	200	5000	648	C0000040	0	0	0
> .pdata	2A00	200	6000	198	40000040	0	0	0
> .src	2C00	200	7000	1E0	40000040	0	0	0
> .reloc	2E00	200	8000	2C	42000040	0	0	0

hello.exe x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0400h:	48	8D	05	29	46	00	00	C3	CC	CC	CC	CC	CC	CC	CC	CC	H..)F..Äïïïïïïïï
0410h:	48	8D	05	11	46	00	00	C3	CC	CC	CC	CC	CC	CC	CC	CC	H...F..Äïïïïïïïï
0420h:	48	89	4C	24	08	48	89	54	24	10	4C	89	44	24	18	4C	H%L\$.H%T\$.L%D\$.L
0430h:	89	4C	24	20	53	56	57	48	83	EC	30	48	8B	F9	48	8D	%L\$ SVWHfï0H<ùH.

位址	十六進位																ASCII															
00000000140001000	48	8D	05	29	46	00	00	C3	CC	CC	CC	CC	CC	CC	CC	CC	H..)F..ÄÄÄ															

PE format

- 但看了一下 .rdata, 好像不是這麼回事?

PE format

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
▼ .text	400	1200	1000	104C	60000020	0	0	0
> .rdata	1600	1200	3000	1140	40000040	0	0	0
> .data	2800	200	5000	648	C0000040	0	0	0
> .pdata	2A00	200	6000	198	40000040	0	0	0
> .rsrc	2C00	200	7000	1E0	40000040	0	0	0
> .reloc	2E00	200	8000	2C	42000040	0	0	0

hello.exe x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1600h:	28	3C	00	00	00	00	00	00	34	3C	00	00	00	00	00	00
1610h:	44	3C	00	00	00	00	00	00	52	3C	00	00	00	00	00	00
1620h:	60	3C	00	00	00	00	00	00	18	41	00	00	00	00	00	00
1630h:	02	41	00	00	00	00	00	00	E8	40	00	00	00	00	00	00

位址	十六進位																ASCII
0000000140003000	D0	4E	0E	38	FB	7E	00	00	30	4E	0E	38	FB	7E	00	00	DO.8û...ON.8û...
0000000140003010	50	4B	0E	38	FB	7E	00	00	E0	48	0E	38	FB	7E	00	00	PK.8û...àH.8û...
0000000140003020	10	4E	0E	38	FB	7E	00	00	B0	01	0E	38	FB	7E	00	00	.N.8û...°.8û...
0000000140003030	A0	EA	51	39	FB	7E	00	00	80	7B	0E	38	FB	7E	00	00	êQ9û...{.8û...

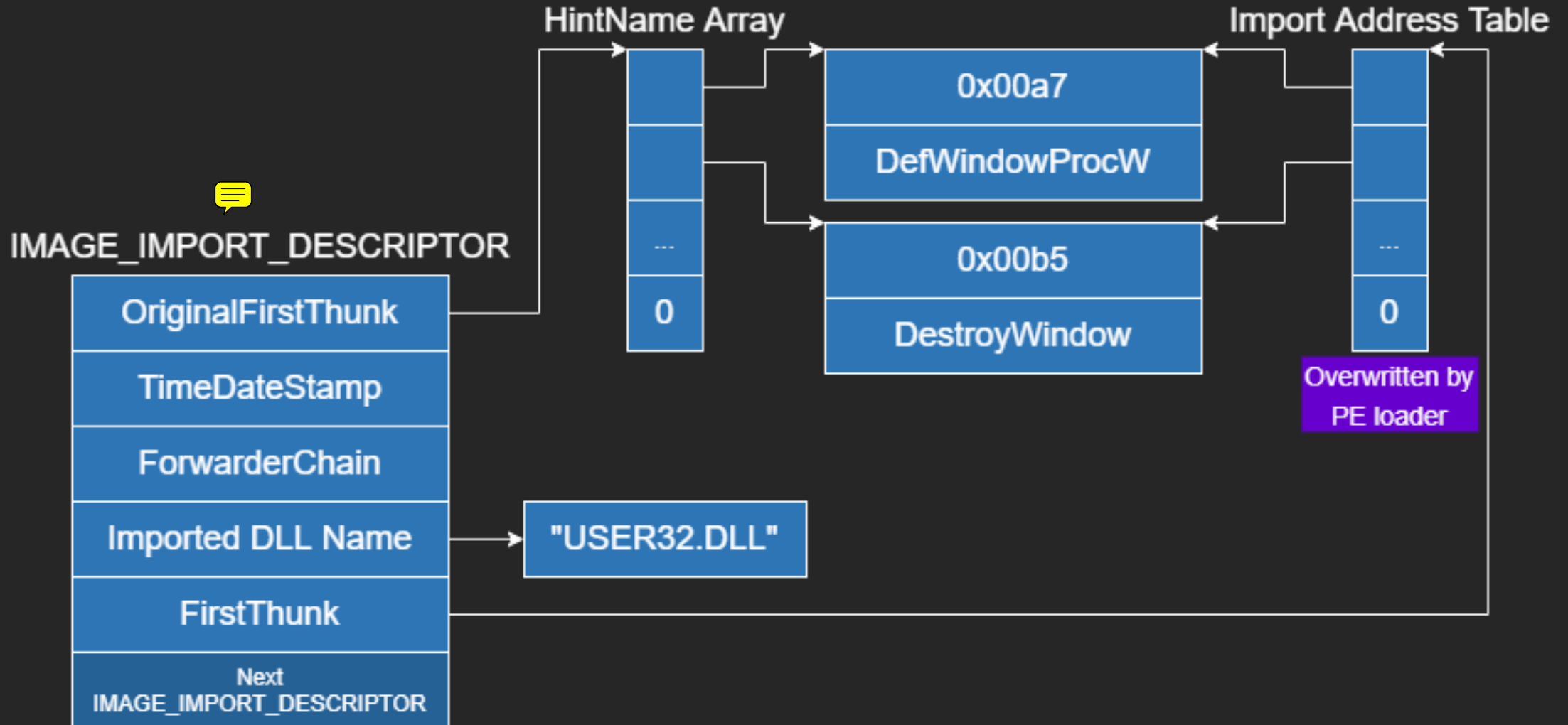
PE format

- 但看了一下 .rdata, 好像不是這麼回事?
- 實際上是因為另一個機制, 改掉了這邊的資料

IAT

Import Address Table

IAT



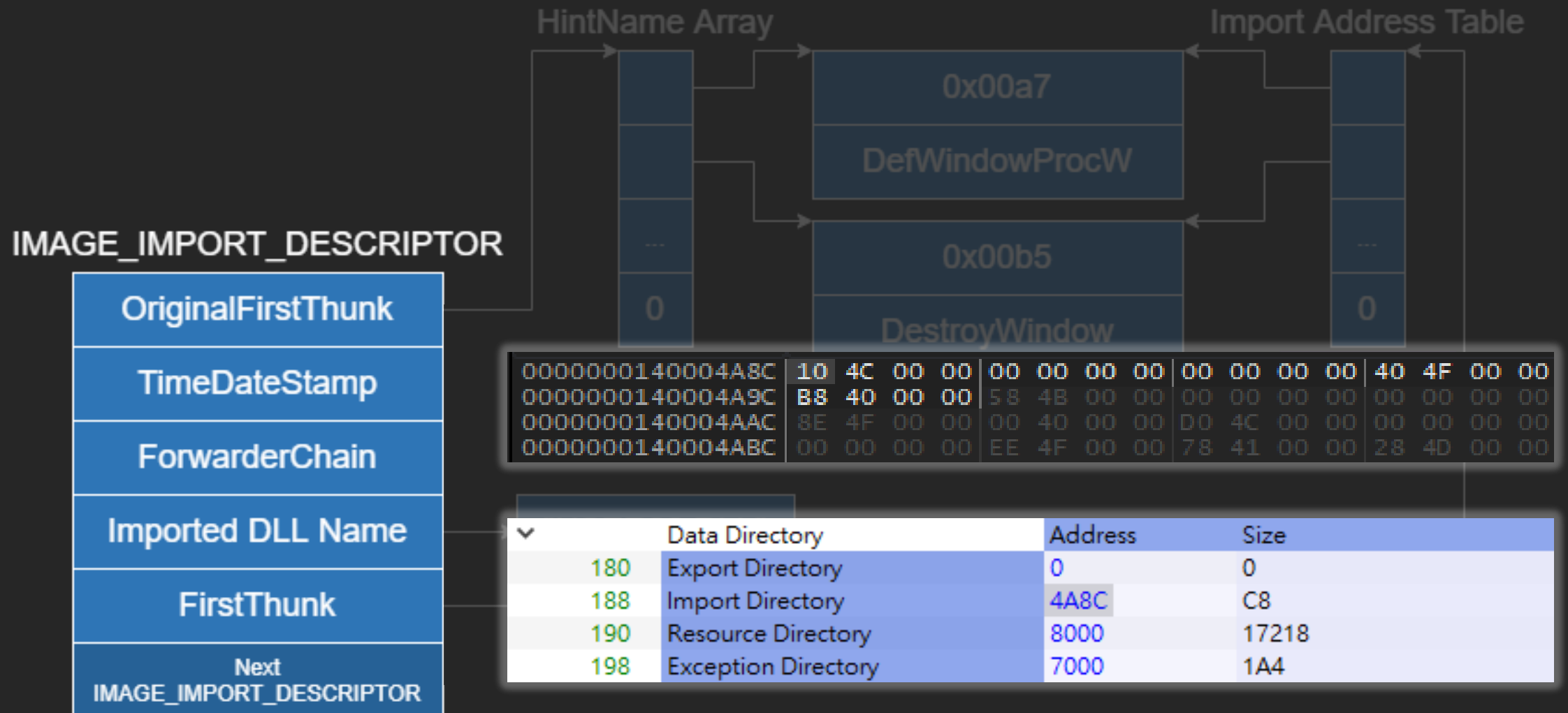
IAT

IMAGE_IMPORT_DESCRIPTOR

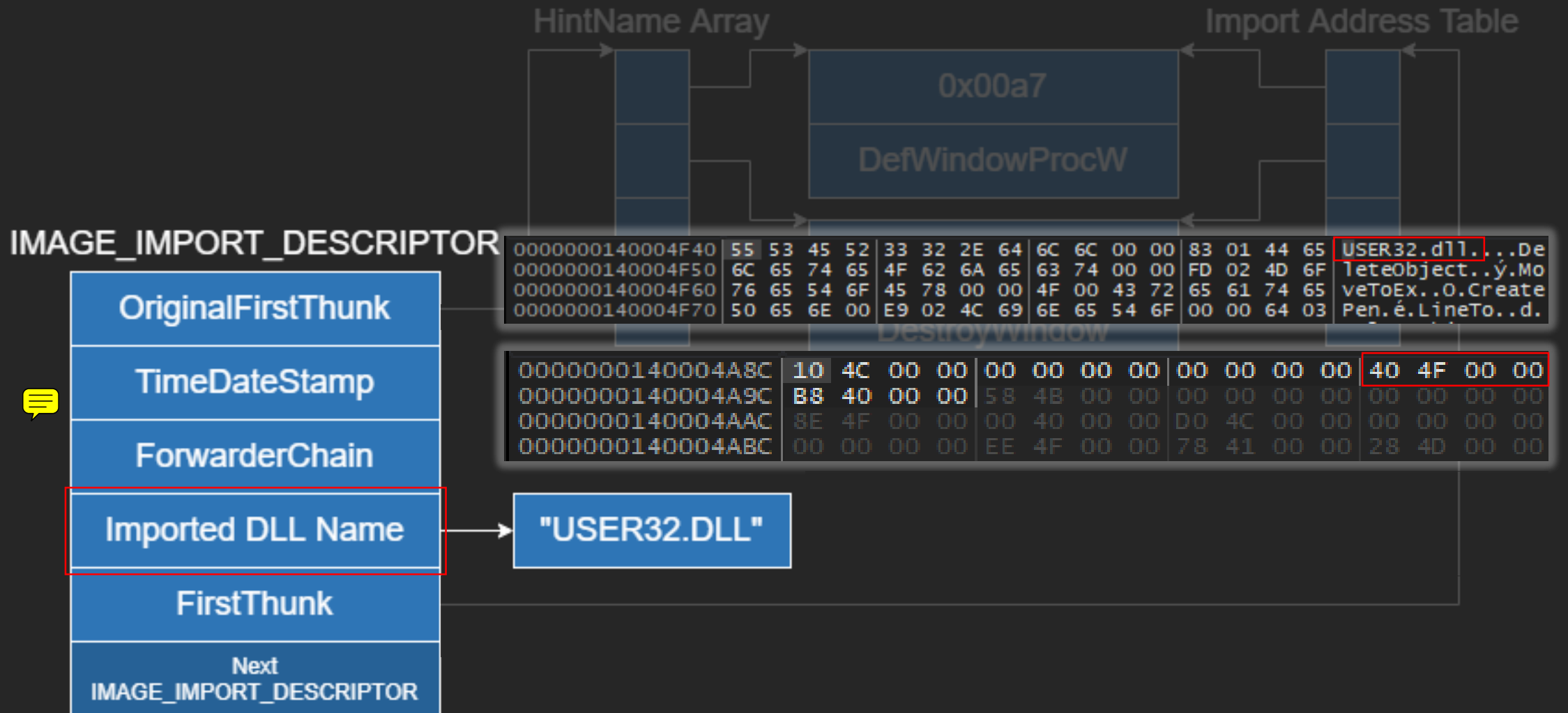
OriginalFirstThunk
TimeDateStamp
ForwarderChain
Imported DLL Name
FirstThunk
Next IMAGE_IMPORT_DESCRIPTOR

Disasm: .rdata	General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section H
Offset	Name	Value	Value			
134	File Alignment	200				
138	OS Ver. (Major)	6	Windows Vista / Server 2008			
13A	OS Ver. (Minor)	0				
13C	Image Ver. (Major)	0				
13E	Image Ver. (Minor)	0				
140	Subsystem Ver. (Major)	6				
142	Subsystem Ver. Minor)	0				
144	Win32 Version Value	0				
148	Size of Image	21000				
14C	Size of Headers	400				
150	Checksum	0				
154	Subsystem	2	Windows GUI			
156	DLL Characteristics	8120				
		100	Image is NX compatible			
		8000	TerminalServer aware			
158	Size of Stack Reserve	100000				
160	Size of Stack Commit	1000				
168	Size of Heap Reserve	100000				
170	Size of Heap Commit	1000				
178	Loader Flags	0				
17C	Number of RVAs and Sizes	10				
	Data Directory	Address	Size			
180	Export Directory	0	0			
188	Import Directory	4A8C	C8			
190	Resource Directory	8000	17218			
198	Exception Directory	7000	1A4			

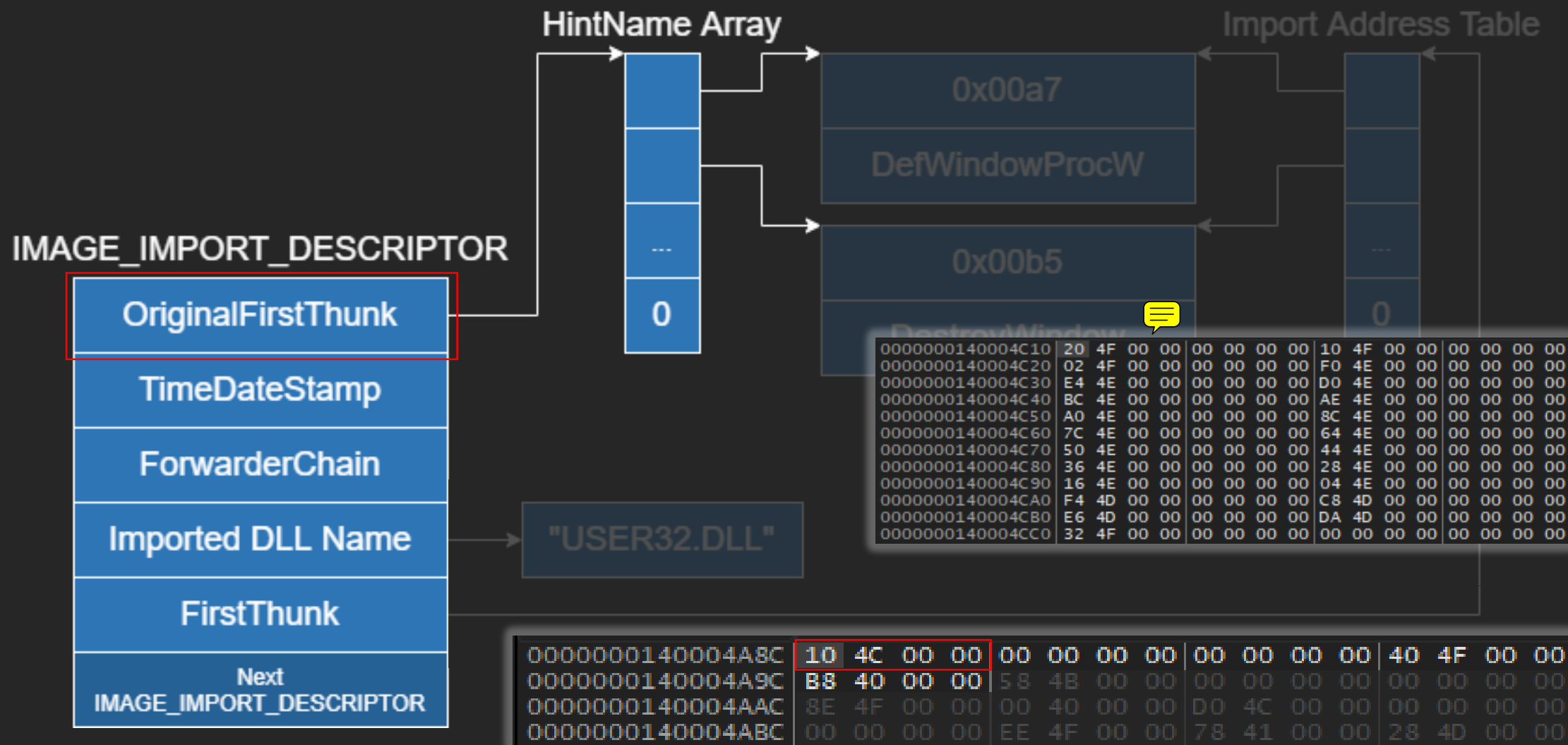
IAT



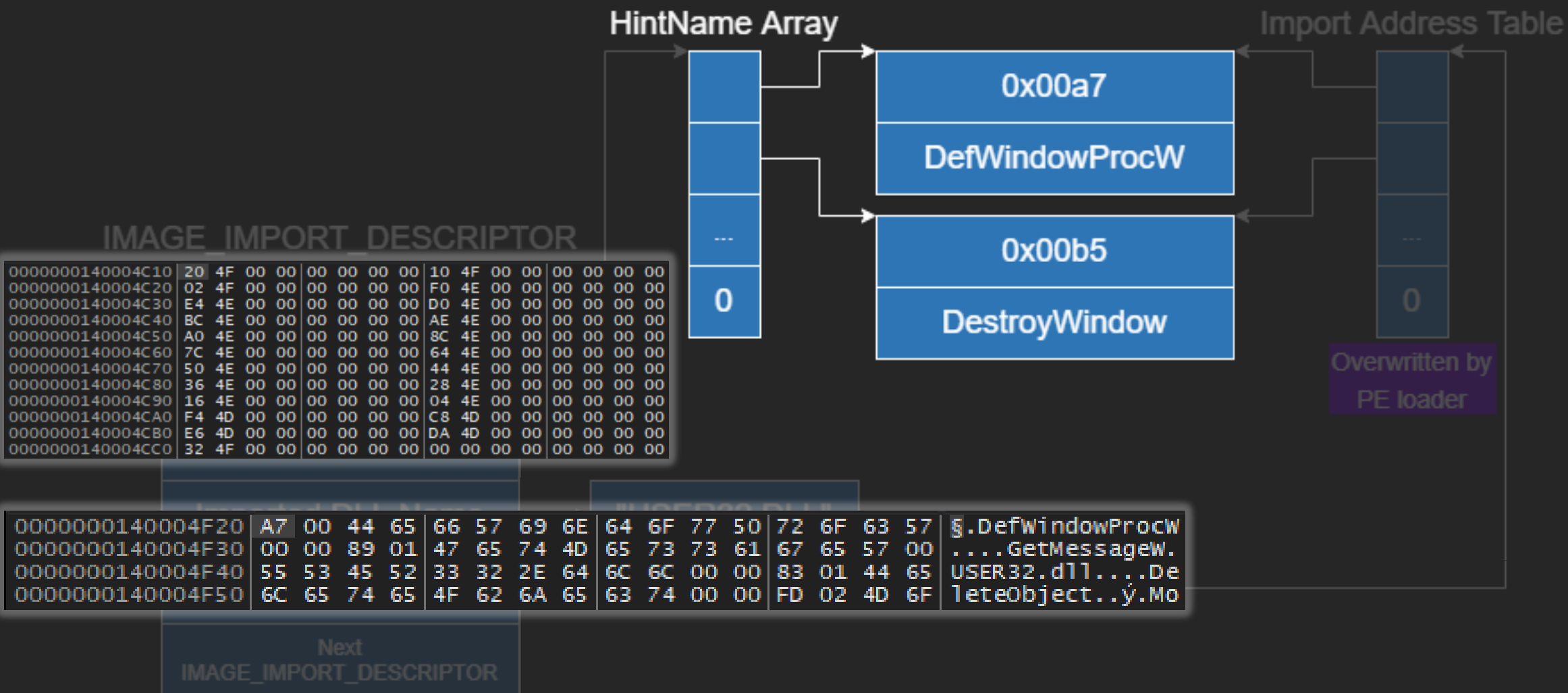
IAT



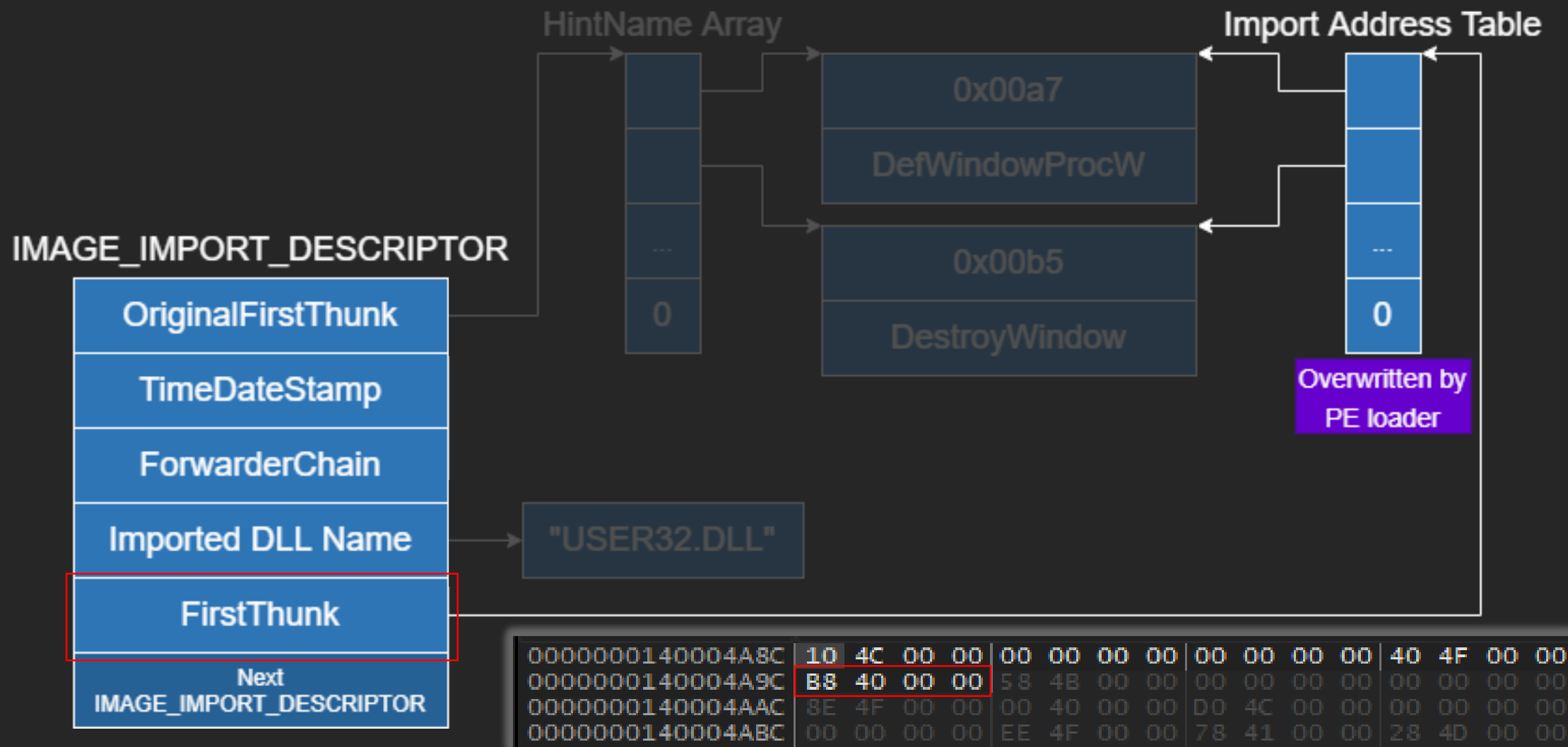
IAT



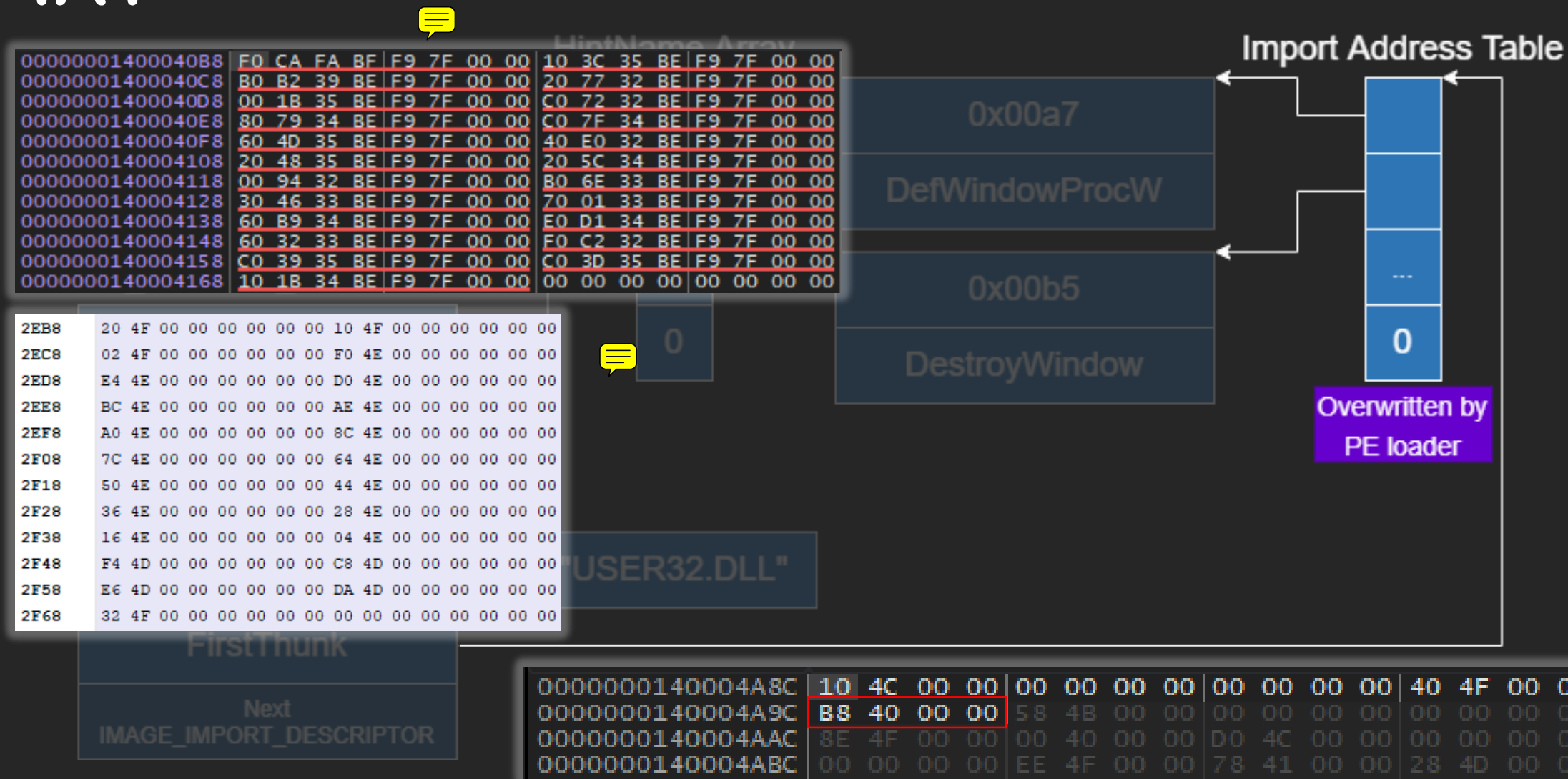
IAT



IAT



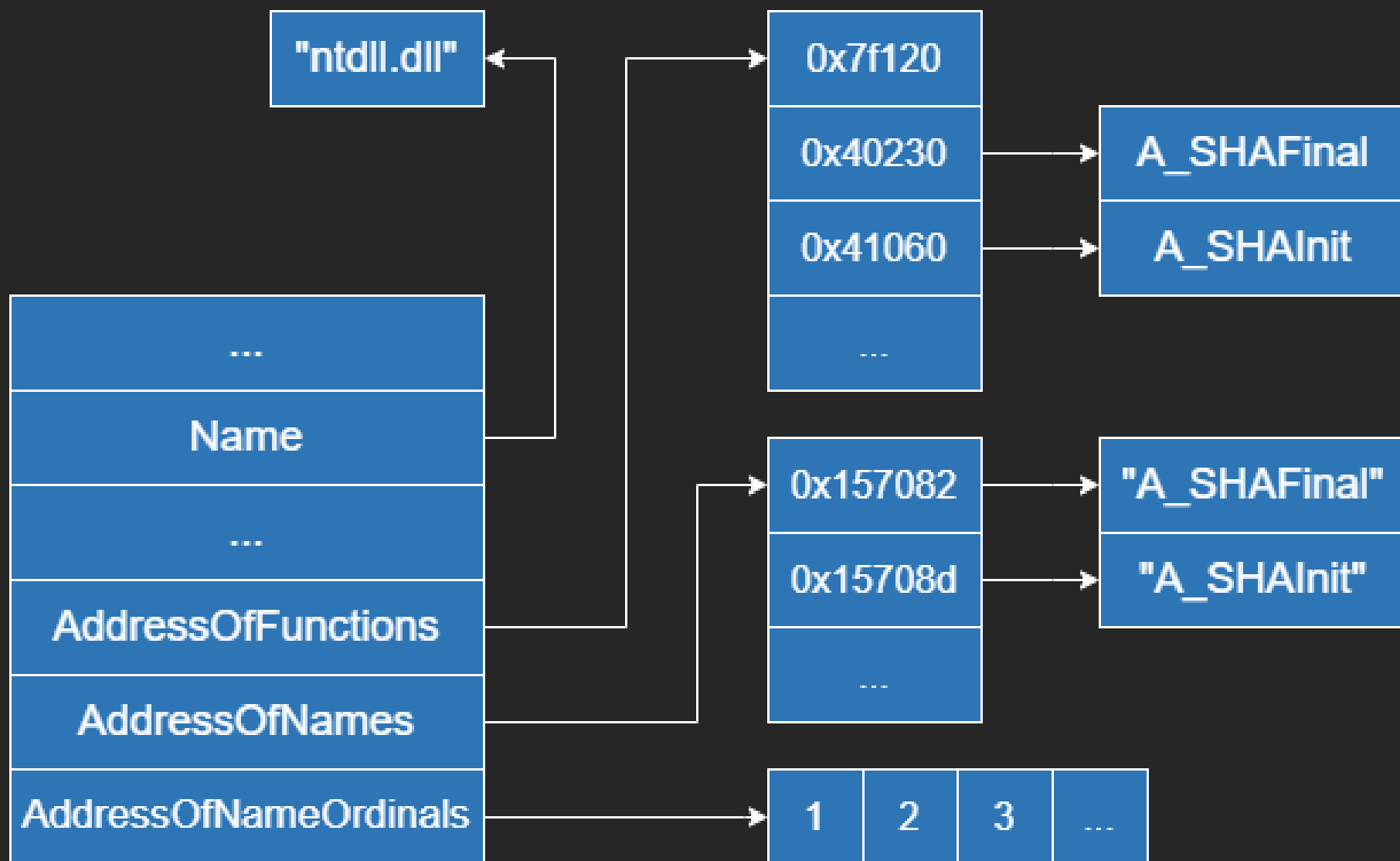
IAT



EAT

Export Address Table

EAT




EAT

"ntdll.dll"

...
Name
...
AddressOfFunctions
AddressOfNames
AddressOfNameOrdinals

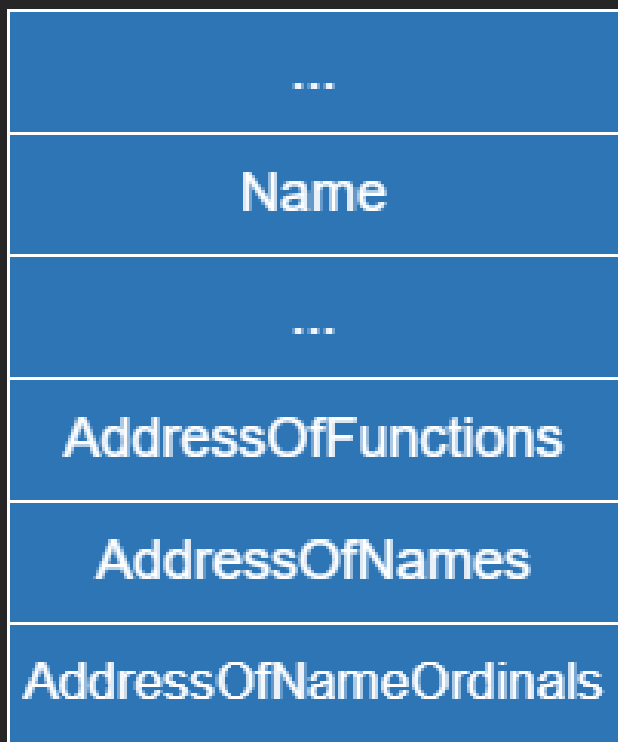
Data Directory	Address	Size
170 Export Directory	151160	12E71

Disasm: .rdata		General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs
+ 							
Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. o
> .text	400	119000	1000	118F1E	60000020	0	0
> PAGE	119400	600	11A000	592	60000020	0	0
> RT	119A00	200	11B000	1F9	60000020	0	0
> .rdata	119C00	48000	11C000	47FD1	40000040	0	0

00007FFE1ED90000	00000000000001000	ntdll.dll
00007FFE1ED91000	000000000000119000	".text"
00007FFE1EEAA000	00000000000001000	"PAGE"
00007FFE1EEAB000	00000000000001000	"RT"
00007FFE1EEAC000	00000000000048000	".rdata"

```
>>> rdata = 0x11C000
>>> rdata_base = 0x0007FFE1EEAC000
>>> export_dir = 0x151160
>>> hex(export_dir - rdata + rdata_base)
'0x7ffe1eee1160'
```

EAT



"ntdll.dll"

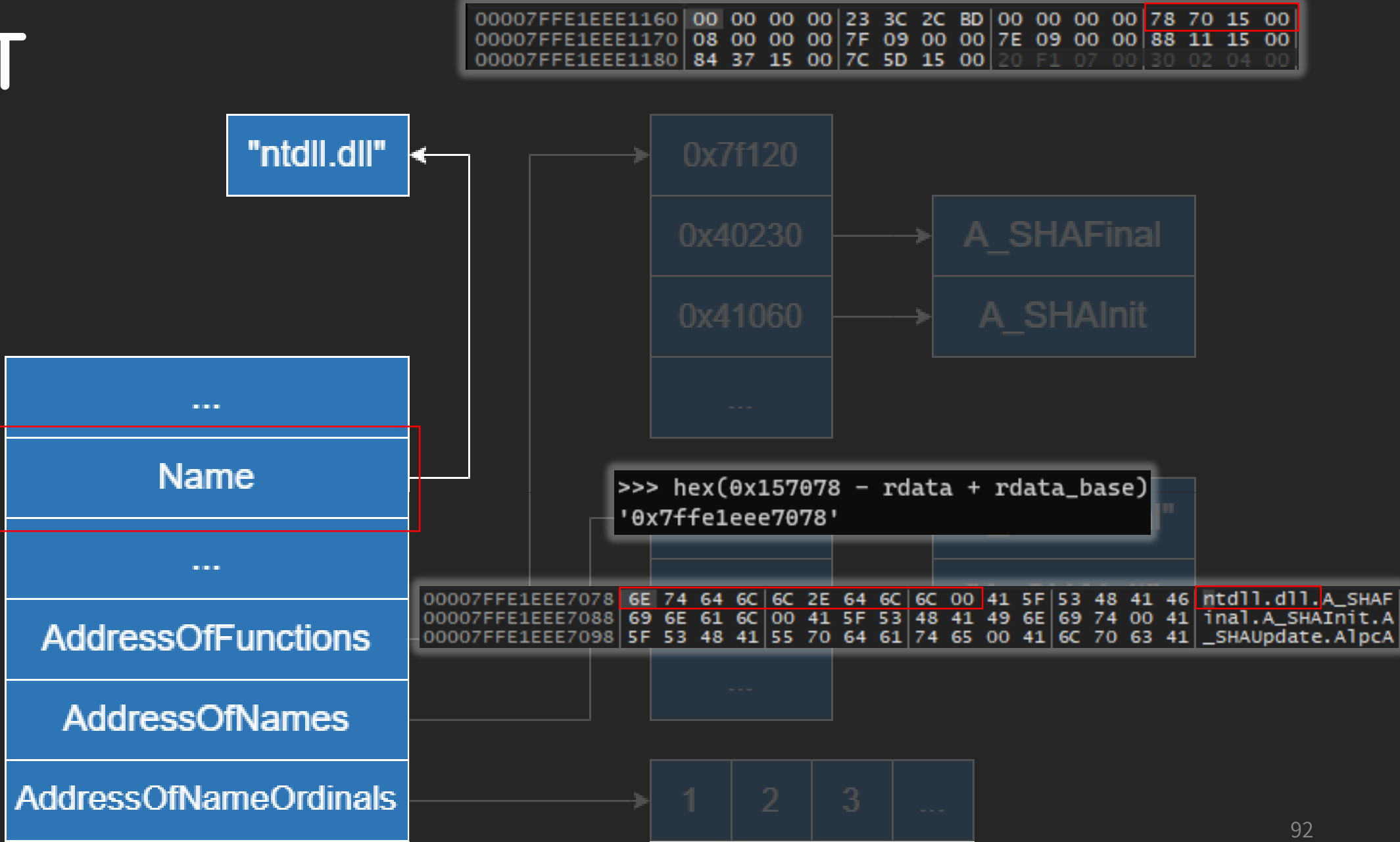
00007FFE1EEE1160	00 00 00 00	23 3C 2C BD	00 00 00 00	78 70 15 00
00007FFE1EEE1170	08 00 00 00	7F 09 00 00	7E 09 00 00	88 11 15 00
00007FFE1EEE1180	84 37 15 00	7C 5D 15 00	20 F1 07 00	30 02 04 00

Offset	Name	Value	Meaning
14ED60	Characteristics	0	
14ED64	TimeDateStamp	BD2C3C23	星期一, 28.07.2070 17:06:43 UTC
14ED68	MajorVersion	0	
14ED6A	MinorVersion	0	
14ED6C	Name	157078	ntdll.dll
14ED70	Base	8	
14ED74	NumberOfFunctions	97F	
14ED78	NumberOfNames	97E	
14ED7C	AddressOfFunctions	151188	
14ED80	AddressOfNames	153784	
14ED84	AddressOfNameOrdinals	155D7C	

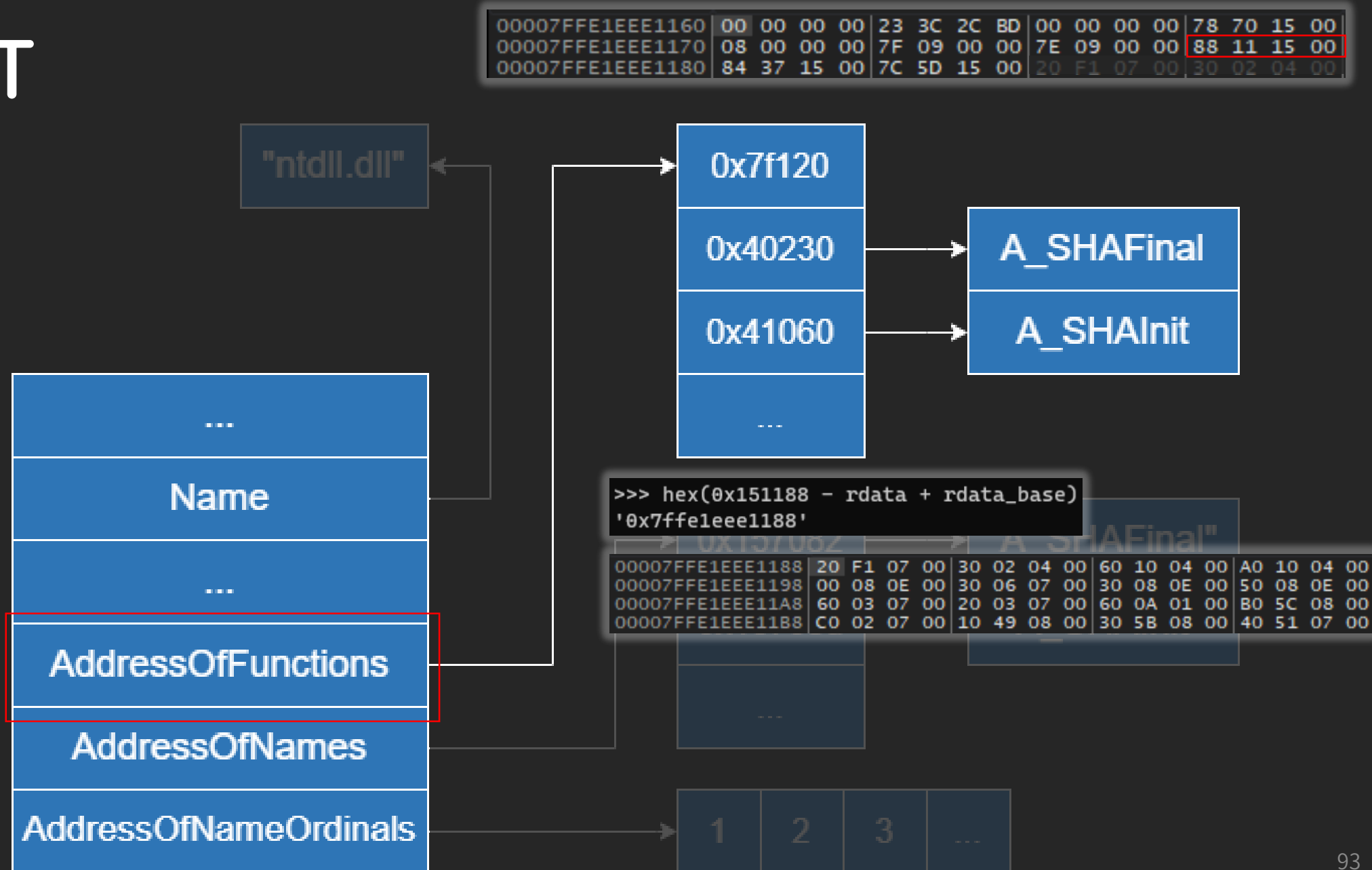


```
>>> rdata = 0x11C000
>>> rdata_base = 0x0007FFE1EEAC000
>>> export_dir = 0x151160
>>> hex(export_dir - rdata + rdata_base)
'0x7ffe1eee1160'
```

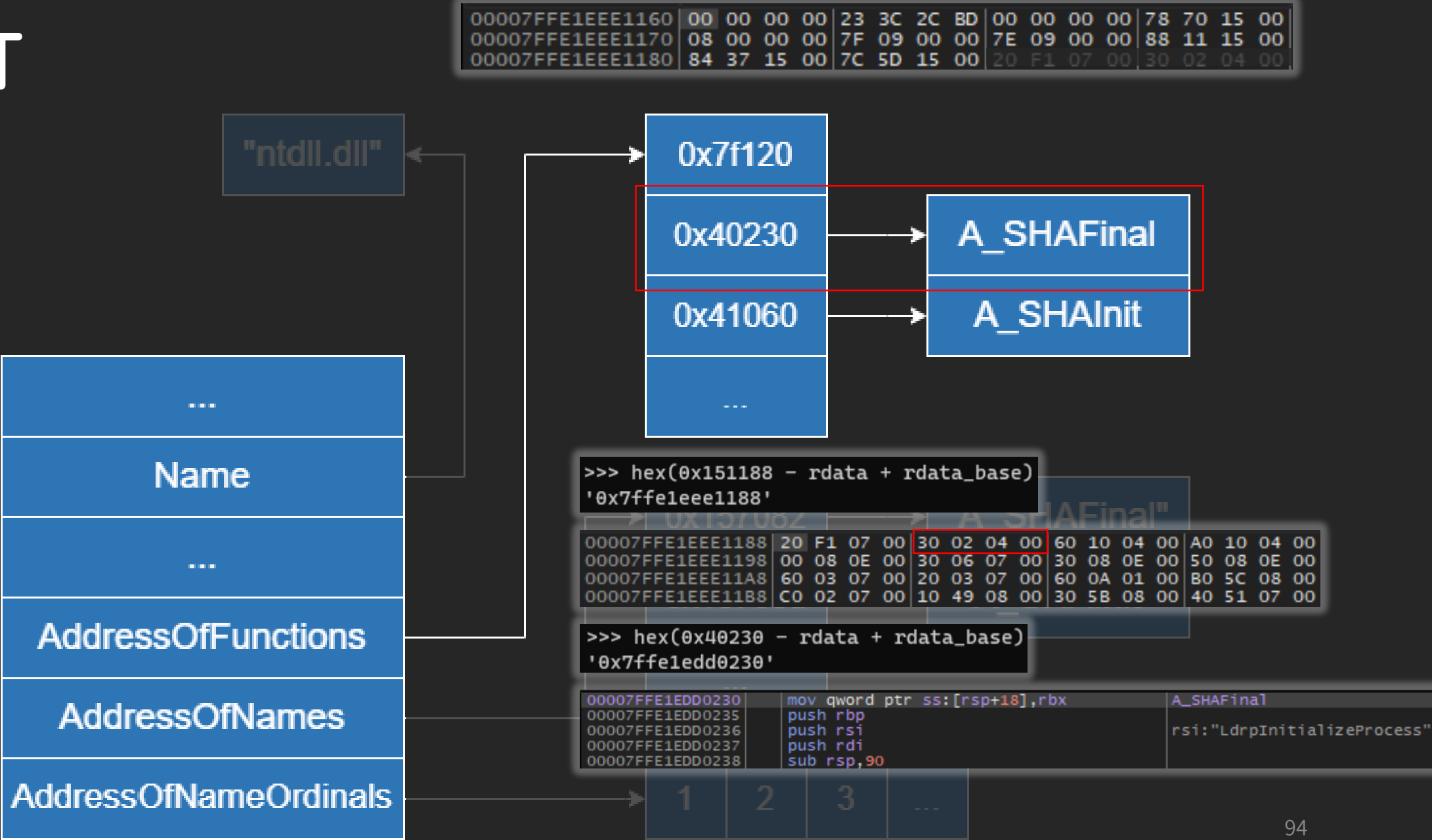
EAT



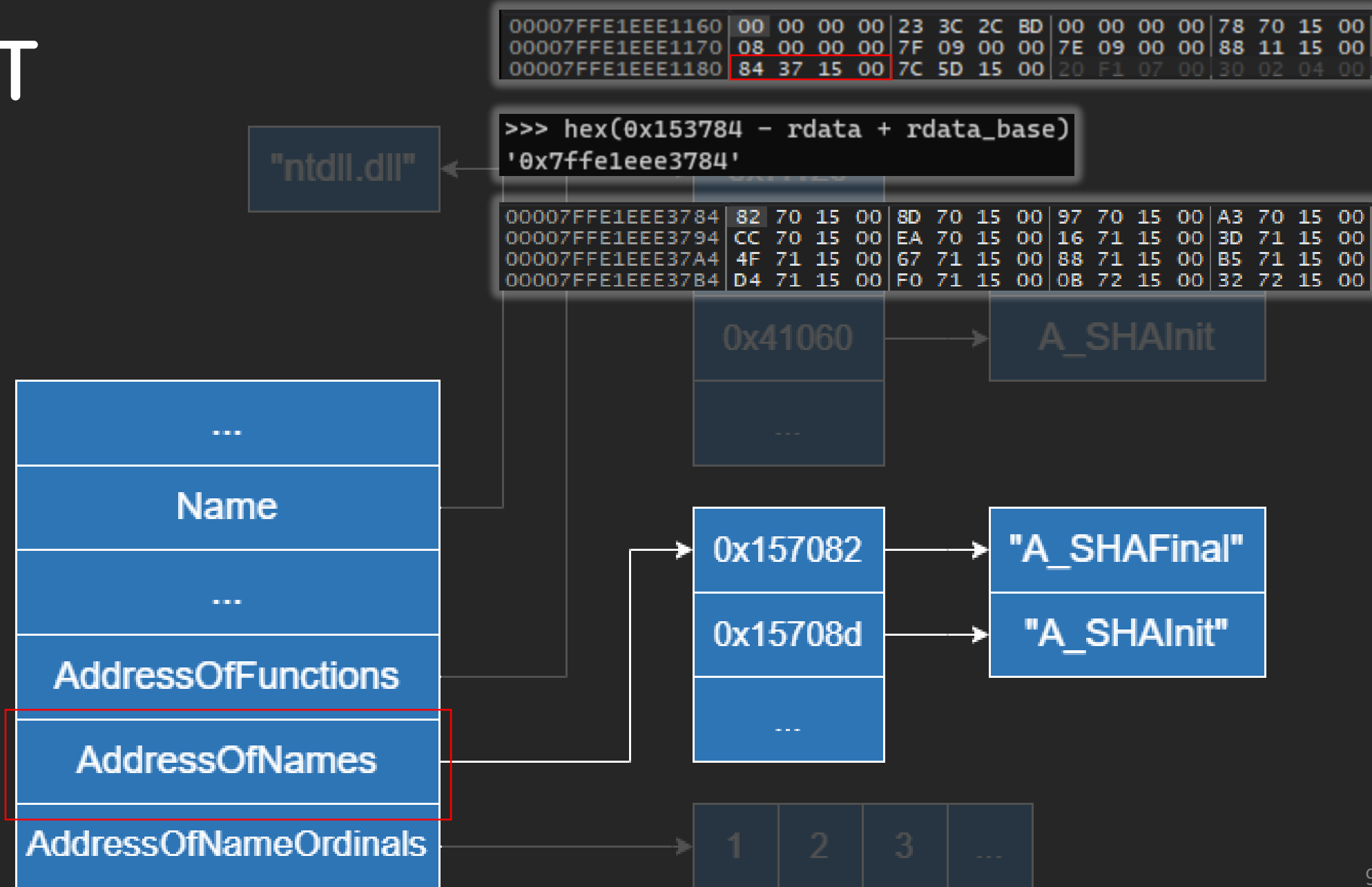
EAT



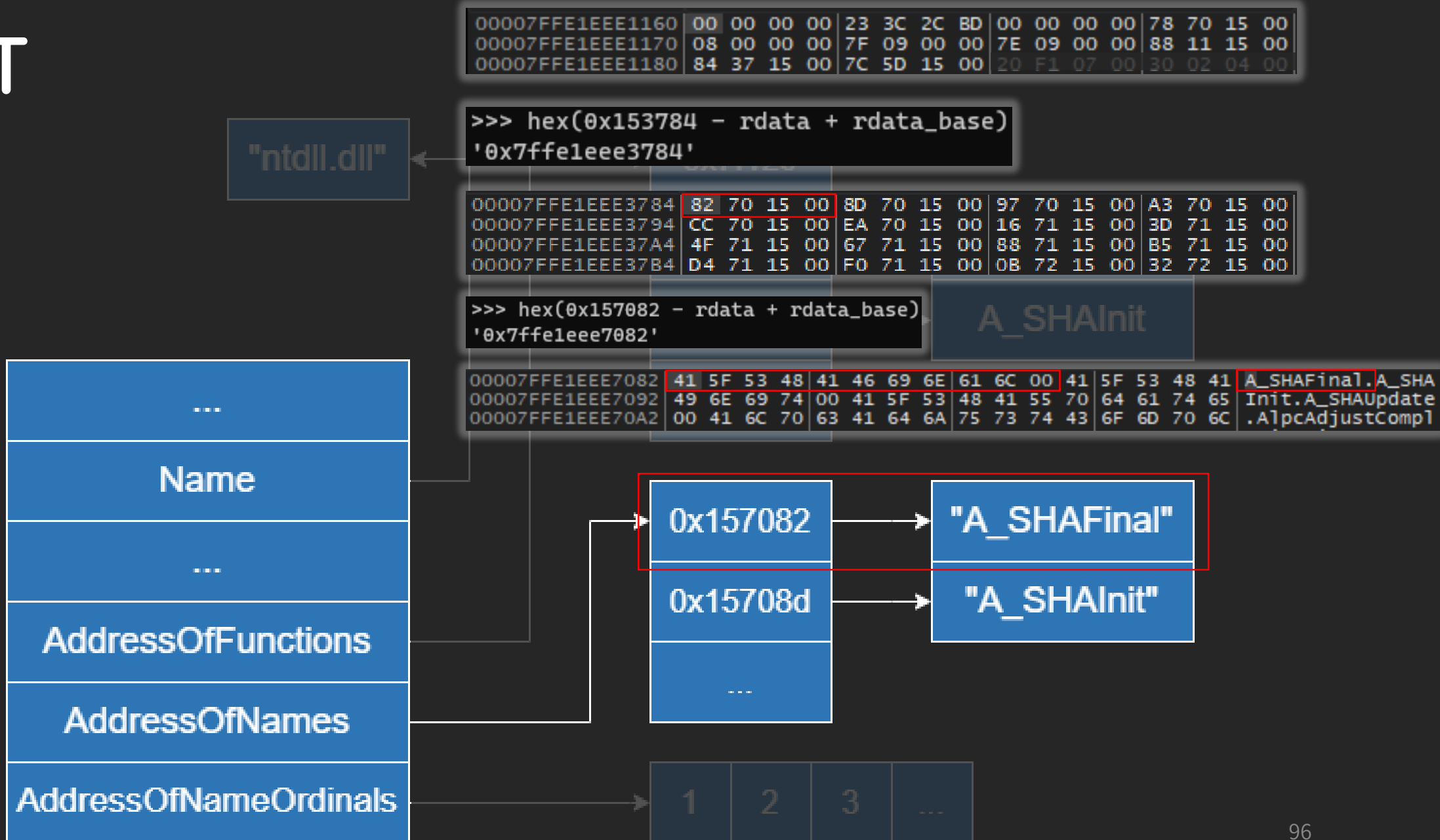
EAT



EAT



EAT

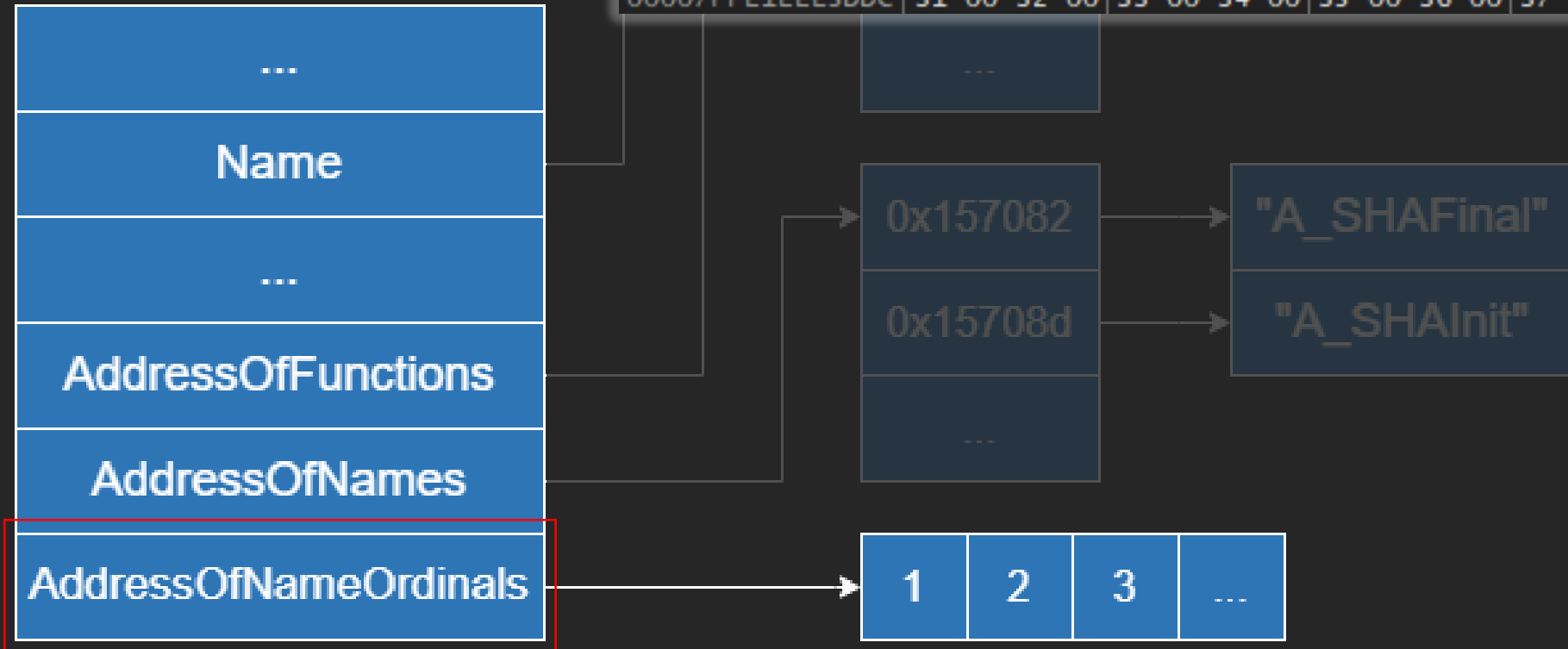


EAT

00007FFE1EEE1160	00	00	00	00	23	3C	2C	BD	00	00	00	00	78	70	15	00
00007FFE1EEE1170	08	00	00	00	7F	09	00	00	7E	09	00	00	88	11	15	00
00007FFE1EEE1180	84	37	15	00	7C	5D	15	00	20	F1	07	00	30	02	04	00

```
>>> hex(0x155d7c - rdata + rdata_base)
'0x7ffe1eee5d7c'
```

00007FFE1EEE5D7C	01	00	02	00	03	00	04	00	05	00	06	00	07	00	08	00
00007FFE1EEE5D8C	09	00	0A	00	0B	00	0C	00	0D	00	0E	00	0F	00	10	00
00007FFE1EEE5D9C	11	00	12	00	13	00	14	00	15	00	16	00	17	00	18	00
00007FFE1EEE5DAC	19	00	1A	00	1B	00	1C	00	1D	00	1E	00	1F	00	20	00
00007FFE1EEE5DBC	21	00	22	00	23	00	24	00	25	00	26	00	27	00	28	00
00007FFE1EEE5DCC	29	00	2A	00	2B	00	2C	00	2D	00	2E	00	2F	00	30	00
00007FFE1EEE5DDC	31	00	32	00	33	00	34	00	35	00	36	00	37	00	38	00



PEB

Process Environment Block

PEB



PEB msdn

- MSDN 裡面是這樣寫的 ...



```
typedef struct _PEB {  
    BYTE Reserved1[2];  
    BYTE BeingDebugged;  
    BYTE Reserved2[1];  
    PVOID Reserved3[2];  
    PPEB_LDR_DATA Ldr;  
    PRTL_USER_PROCESS_PARAMETERS ProcessParameters;  
    PVOID Reserved4[3];  
    PVOID AtlThunkSListPtr;  
    PVOID Reserved5;  
    ULONG Reserved6;  
    PVOID Reserved7;  
    ULONG Reserved8;  
    ULONG AtlThunkSListPtr32;  
    PVOID Reserved9[45];  
    BYTE Reserved10[96];  
    PPS_POST_PROCESS_INIT_ROUTINE PostProcessInitRoutine;  
    BYTE Reserved11[128];  
    PVOID Reserved12[1];  
    ULONG SessionId;  
} PEB, *PPEB;
```

PEB

- 改用 windbg 來查看...

```
||1:1:001> dt _peb @$peb
ntdll!_PEB
+0x000 InheritedAddressSpace : 0 ''
+0x001 ReadImageFileExecOptions : 0 ''
+0x002 BeingDebugged : 0x1 ''
+0x003 BitField : 0 ''
+0x003 ImageUsesLargePages : 0y0
+0x003 IsProtectedProcess : 0y0
+0x003 IsImageDynamicallyRelocated : 0y0
+0x003 SkipPatchingUser32Forwarders : 0y0
+0x003 IsPackagedProcess : 0y0
+0x003 IsAppContainer : 0y0
+0x003 IsProtectedProcessLight : 0y0
+0x003 IsLongPathAwareProcess : 0y0
+0x004 Padding0 : [4] ""
+0x008 Mutant : 0xffffffff`ffffffff Void
+0x010 ImageBaseAddress : 0x00000000`00400000 Void
+0x018 Ldr : 0x000007fff`dd0c53c0 _PEB_LDR_DATA
+0x020 ProcessParameters : 0x00000000`00772190 _RTL_USER_PROCESS_PARAMETERS
+0x028 SubSystemData : (null)
+0x030 ProcessHeap : 0x00000000`00770000 Void
+0x038 FastPebLock : 0x000007fff`dd0c4fe0 _RTL_CRITICAL_SECTION
+0x040 AtlThunkSListPtr : (null)
+0x048 IFE0Key : (null)
+0x050 CrossProcessFlags : 2
+0x050 ProcessInJob : 0y0
+0x050 ProcessInitializing : 0y1
+0x050 ProcessUsingVEH : 0y0
+0x050 ProcessUsingVCH : 0y0
+0x050 ProcessUsingFTH : 0y0
+0x050 ProcessPreviouslyThrottled : 0y0
+0x050 ProcessCurrentlyThrottled : 0y0
+0x050 ProcessImagesHotPatched : 0y0
+0x050 ReservedBits0 : 0y00000000000000000000000000000000 (0)
+0x054 Padding1 : [4] ""
+0x058 KernelCallbackTable : (null)
+0x058 UserSharedInfoPtr : (null)
+0x060 SystemReserved : 0
+0x064 AtlThunkSListPtr32 : 0
```



MSDN

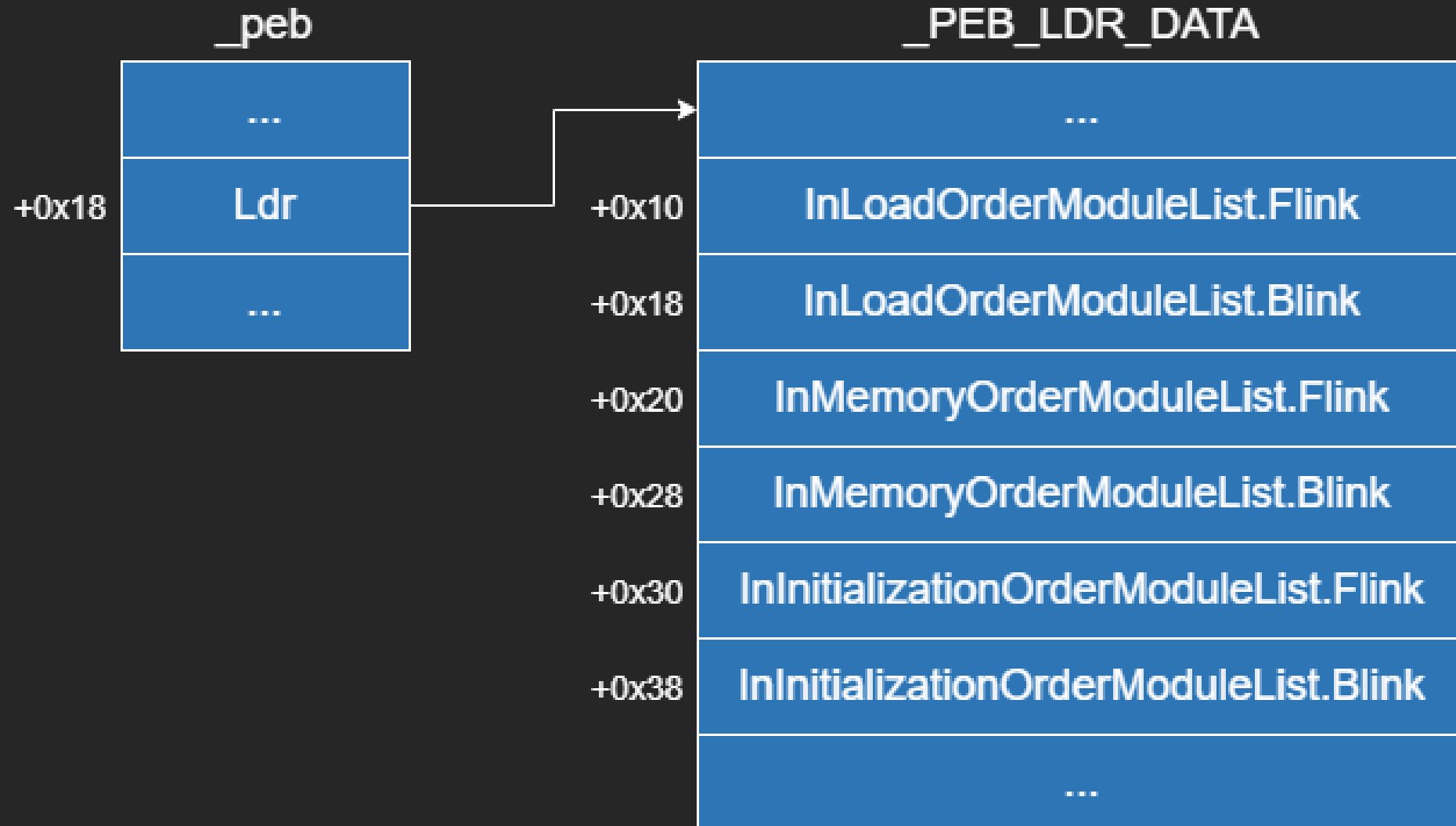
騙你的

PEB

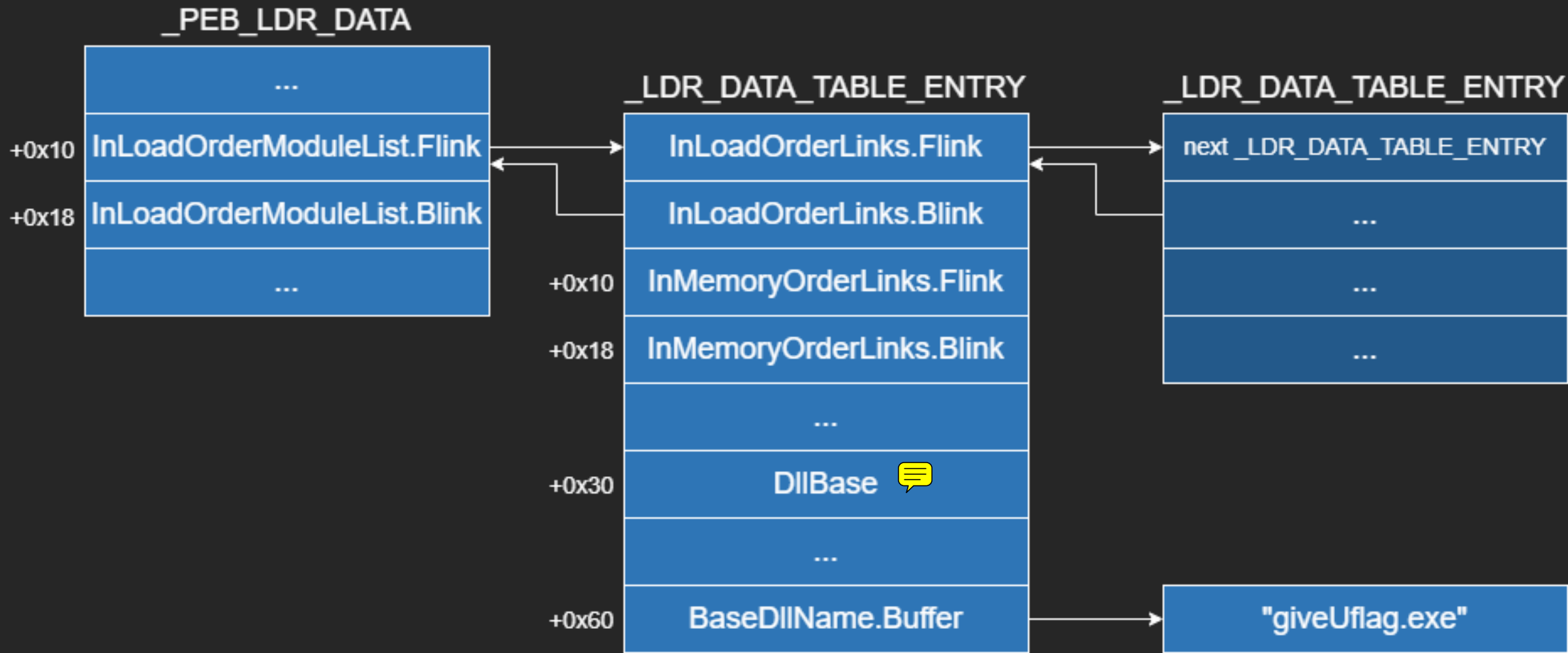
- 取得 PEB
- 32bit
 - FS:[0x30]
- 64bit
 - GS:[0x60]

```
mov     rax, gs:60h      ; get PEB
mov     [rbp+PEB], rax
```

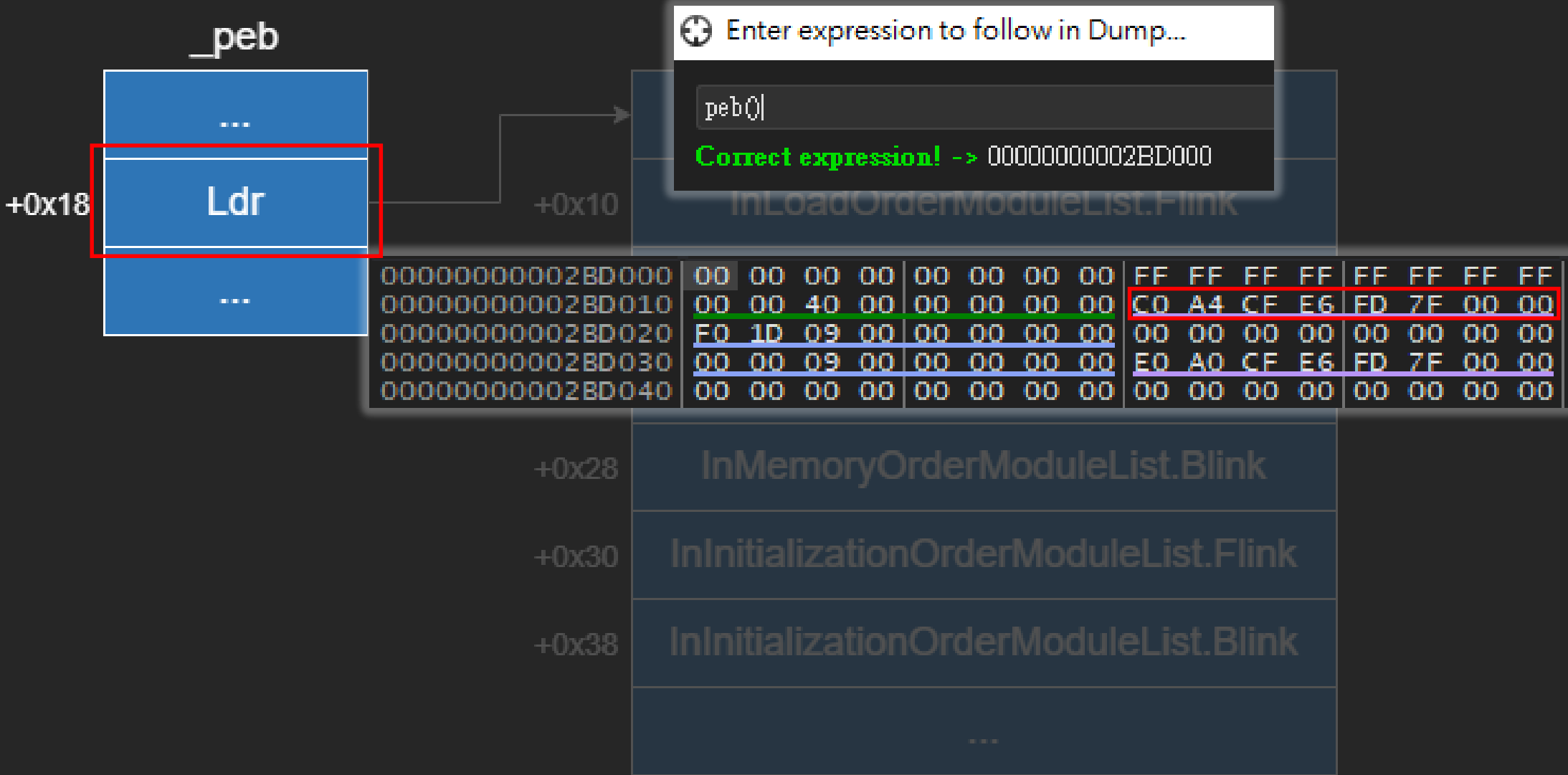
PEB



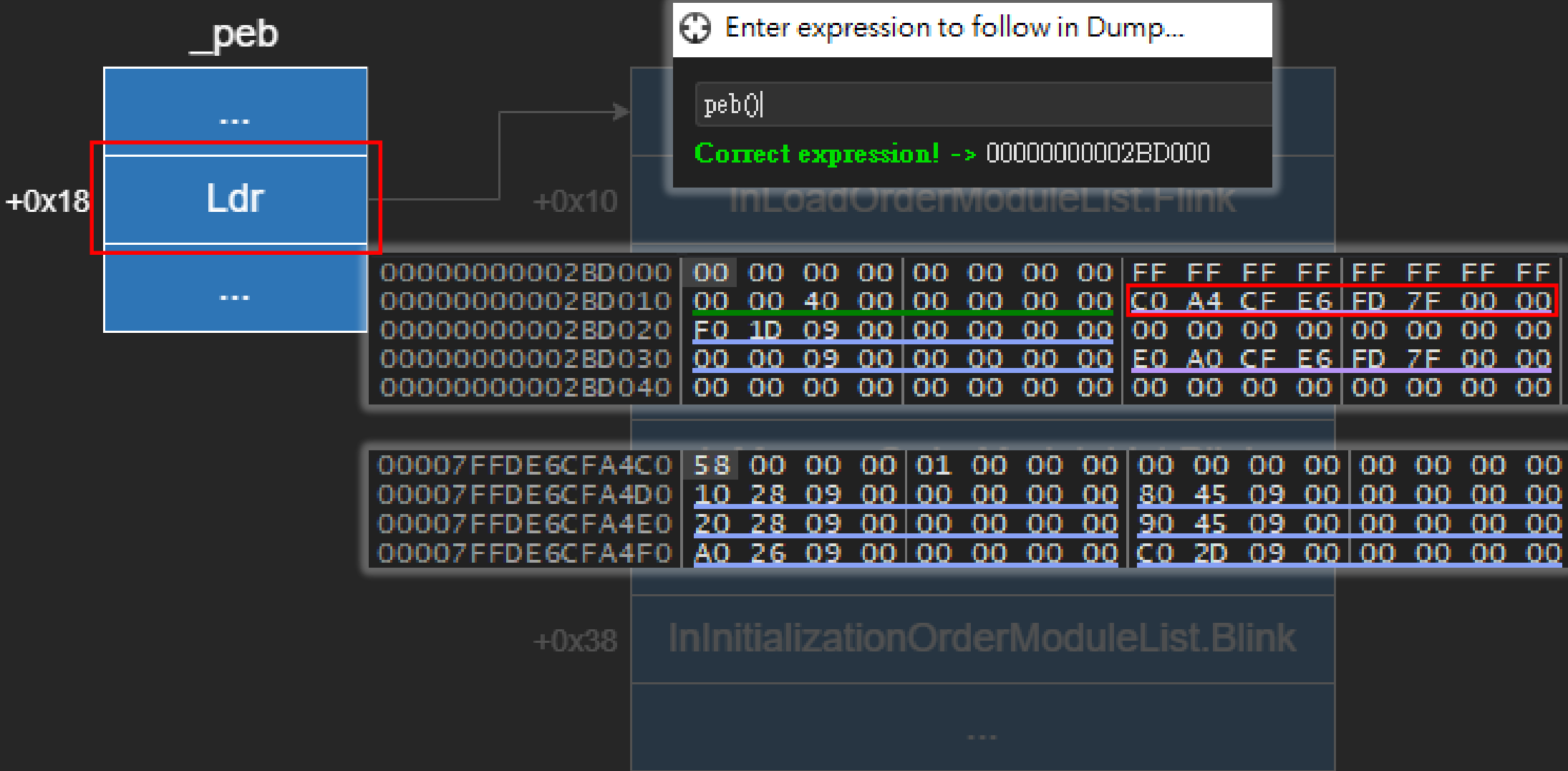
PEB



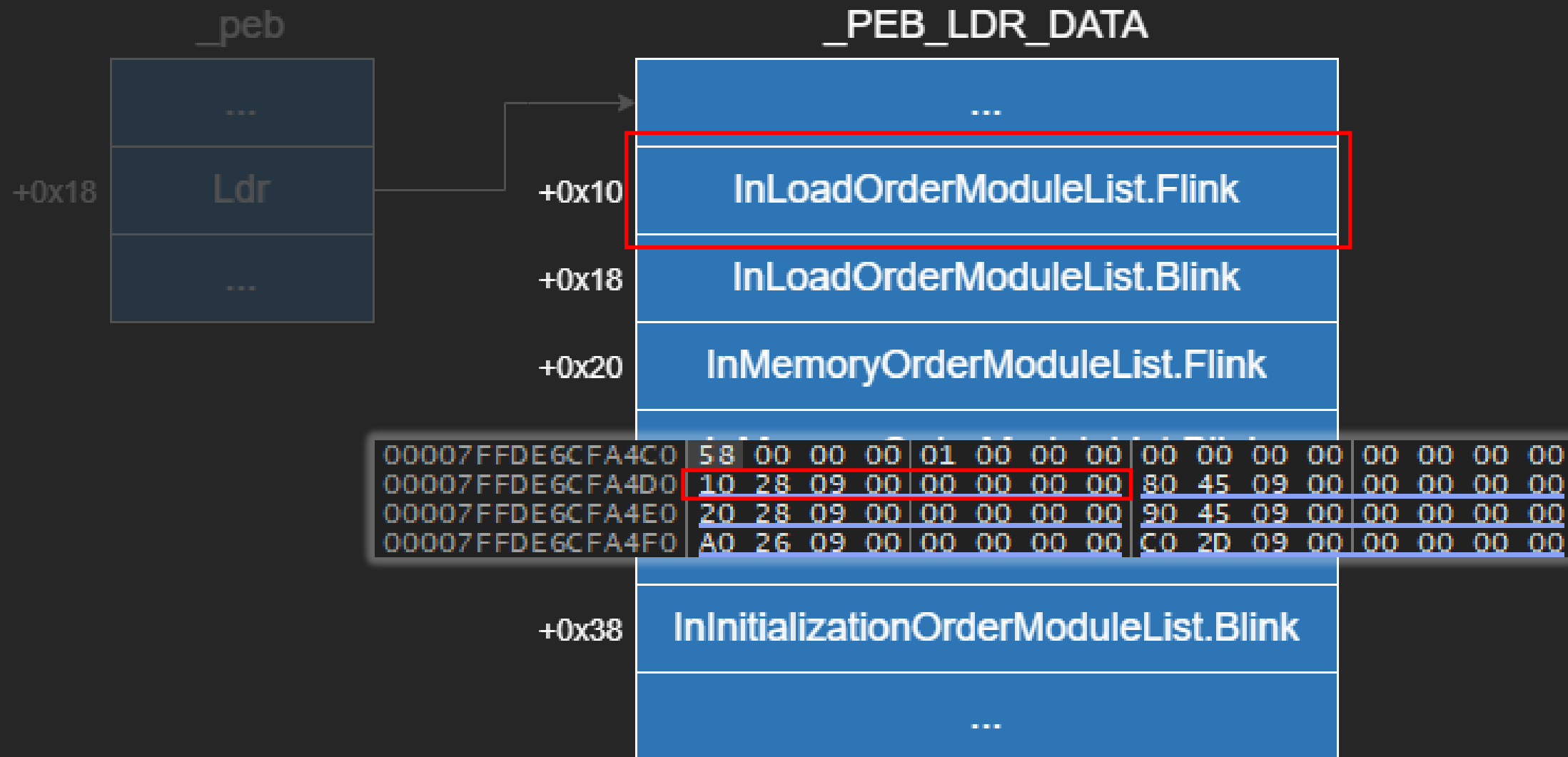
PEB



PEB



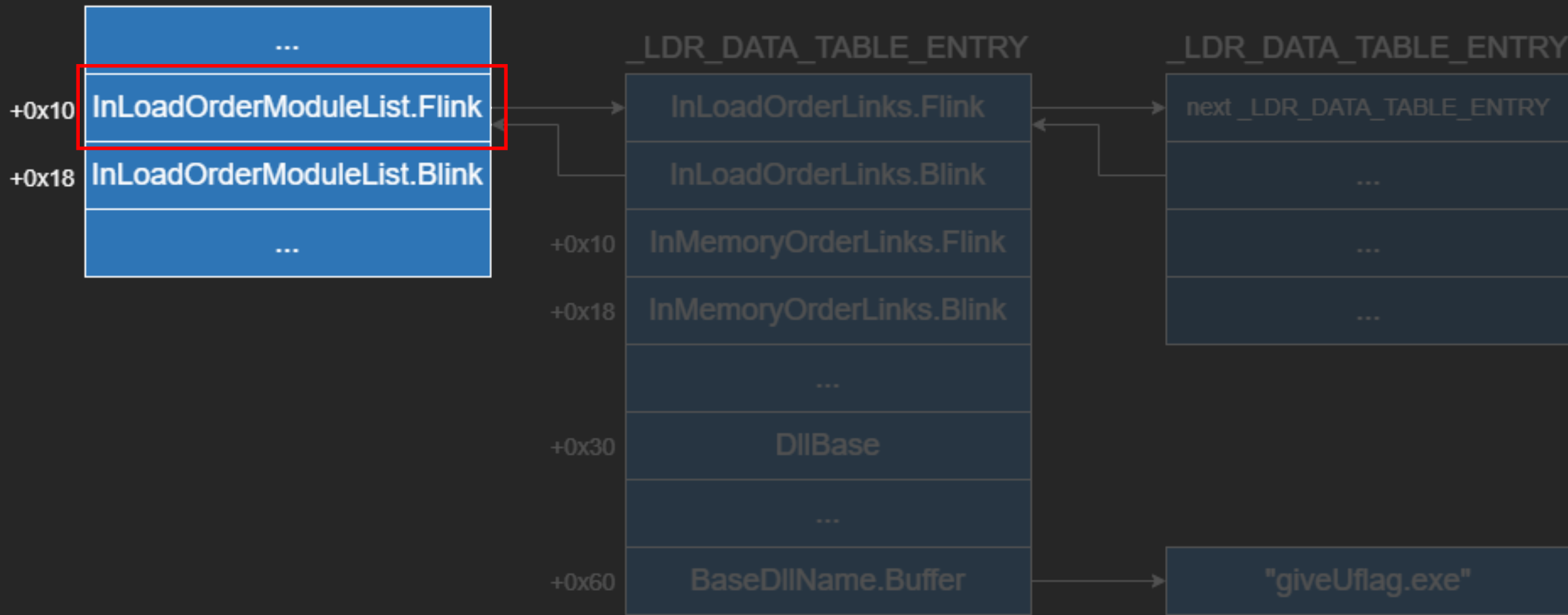
PEB



PEB

_PEB_LDR_DATA

00007FFDE6CFA4C0	58	00	00	00	01	00	00	00	00	00	00	00	00	00
00007FFDE6CFA4D0	10	28	09	00	00	00	00	00	80	45	09	00	00	00
00007FFDE6CFA4E0	20	28	09	00	00	00	00	00	90	45	09	00	00	00
00007FFDE6CFA4F0	A0	26	09	00	00	00	00	00	C0	2D	09	00	00	00



PEB

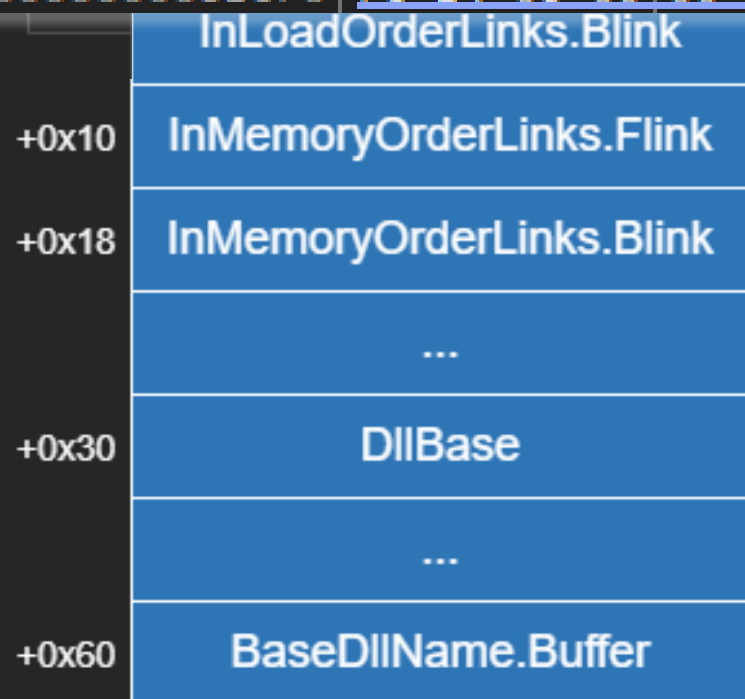
00007FFDE6CFA4C0	58	00	00	00	01	00	00	00	00	00	00	00	00	00	00
00007FFDE6CFA4D0	10	28	09	00	00	00	00	00	80	45	09	00	00	00	00
00007FFDE6CFA4E0	20	28	09	00	00	00	00	00	90	45	09	00	00	00	00
00007FFDE6CFA4F0	A0	26	09	00	00	00	00	00	C0	2D	09	00	00	00	00

_PEB_LDR_DATA

00000000000092810	80	26	09	00	00	00	00	00	D0	A4	CF	E6	FD	7F	00	00
00000000000092820	90	26	09	00	00	00	00	00	E0	A4	CF	E6	FD	7F	00	00
00000000000092830	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000000092840	00	00	40	00	00	00	00	00	E0	14	40	00	00	00	00	00
00000000000092850	00	80	00	00	00	00	00	00	5A	00	5C	00	00	00	00	00
00000000000092860	38	24	09	00	00	00	00	00	1A	00	1C	00	00	00	00	00
00000000000092870	78	24	09	00	00	00	00	00	CC	22	00	00	FF	FF	FF	FF

+0x10 InLoadOrderModuleList.Flink

+0x18 InLoadOrderModuleList.Blink



PEB

00007FFDE6CFA4C0	58	00	00	00	01	00	00	00	00	00	00	00	00	00	00
00007FFDE6CFA4D0	10	28	09	00	00	00	00	00	80	45	09	00	00	00	00
00007FFDE6CFA4E0	20	28	09	00	00	00	00	00	90	45	09	00	00	00	00
00007FFDE6CFA4F0	A0	26	09	00	00	00	00	00	C0	2D	09	00	00	00	00

_PEB_LDR_DATA

00000000000092810	80	26	09	00	00	00	00	00	D0	A4	CF	E6	FD	7F	00	00
00000000000092820	90	26	09	00	00	00	00	00	E0	A4	CF	E6	FD	7F	00	00
00000000000092830	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000000092840	00	00	40	00	00	00	00	00	E0	14	40	00	00	00	00	00
00000000000092850	00	80	00	00	00	00	00	00	5A	00	5C	00	00	00	00	00
00000000000092860	38	24	09	00	00	00	00	00	1A	00	1C	00	00	00	00	00
00000000000092870	78	24	09	00	00	00	00	00	CC	22	00	00	FF	FF	FF	FF

+0x10

InLoadOrderModuleList.Flink

+0x18

InLoadOrderModuleList.Blink

InLoadOrderLinks.Blink

+0x10

InMemoryOrderLinks.Flink

+0x18

InMemoryOrderLinks.Blink

...

+0x30

DllBase

...

+0x60

BaseDllName.Buffer

"giveUflag.exe"

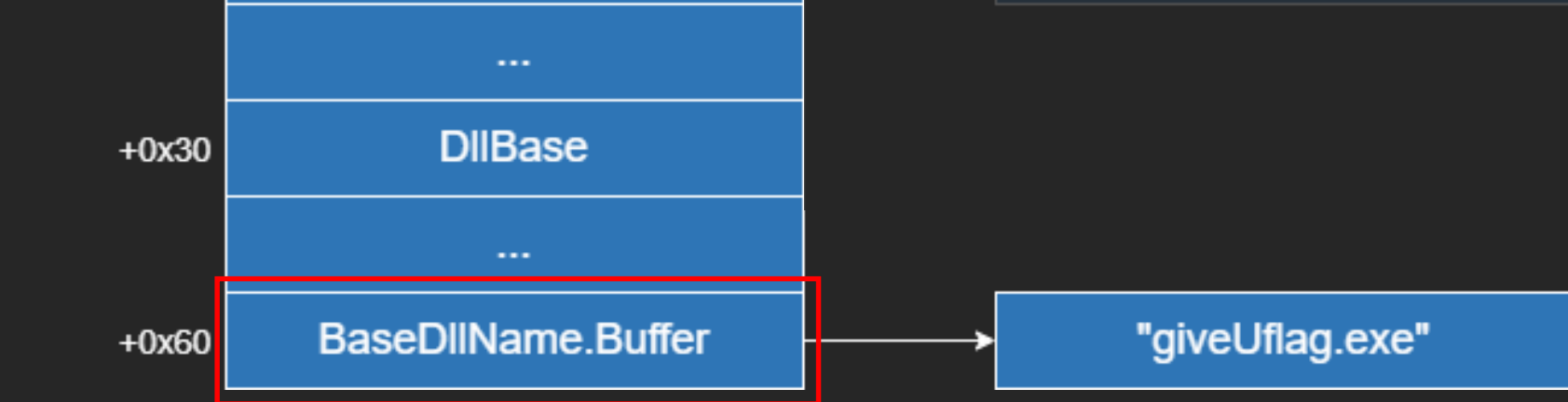
PEB

00007FFDE6CFA4C0	58	00	00	00	01	00	00	00	00	00	00	00	00	00	00
00007FFDE6CFA4D0	10	28	09	00	00	00	00	00	80	45	09	00	00	00	00
00007FFDE6CFA4E0	20	28	09	00	00	00	00	00	90	45	09	00	00	00	00
00007FFDE6CFA4F0	A0	26	09	00	00	00	00	00	C0	2D	09	00	00	00	00

00000000000092810	80	26	09	00	00	00	00	00	D0	A4	CF	E6	FD	7F	00
00000000000092820	90	26	09	00	00	00	00	00	E0	A4	CF	E6	FD	7F	00
00000000000092830	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000000092840	00	00	40	00	00	00	00	00	E0	14	40	00	00	00	00
00000000000092850	00	80	00	00	00	00	00	00	5A	00	5C	00	00	00	00
00000000000092860	38	24	09	00	00	00	00	00	1A	00	1C	00	00	00	00
00000000000092870	78	24	09	00	00	00	00	00	CC	22	00	00	FF	FF	FF

00000000000092478	67	00	69	00	76	00	65	00	55	00	66	00	6C	00	61
00000000000092488	67	00	2E	00	65	00	78	00	65	00	00	00	22	00	43

00000000000092478	67	00	69	00	76	00	65	00	55	00	66	00	6C	00	61
00000000000092488	67	00	2E	00	65	00	78	00	65	00	00	00	22	00	43



PEB

- 一直順著 linked list 爬下去
- 檢查 BaseDllName 是否為想用的模組 (dll / exe)
- 通過 DllBase 取得該模組的 base address
- 解析在 base address 的 PE format, 從 EAT 爬到想用的 API
- Manual Symbol Resolution!



Q & A

下課囉 \(. _ .)>