solarwinds msp

Functions hooking for Windows in C/C++ applications

Andrew Yakimov, developer. 2018
Email: a.yakimov@mail.com

Report plan

solarwinds msp

- Introduction to hooks
- Scope of applications
- Usage examples
- Basic dll injection methods
- Portable Executable
- Main hooks overview
- Windows Native Api Overview
- System call instructions
- Pros and cons
- Antivirus reaction
- Useful links

Introduction to hooks

solarwinds msp

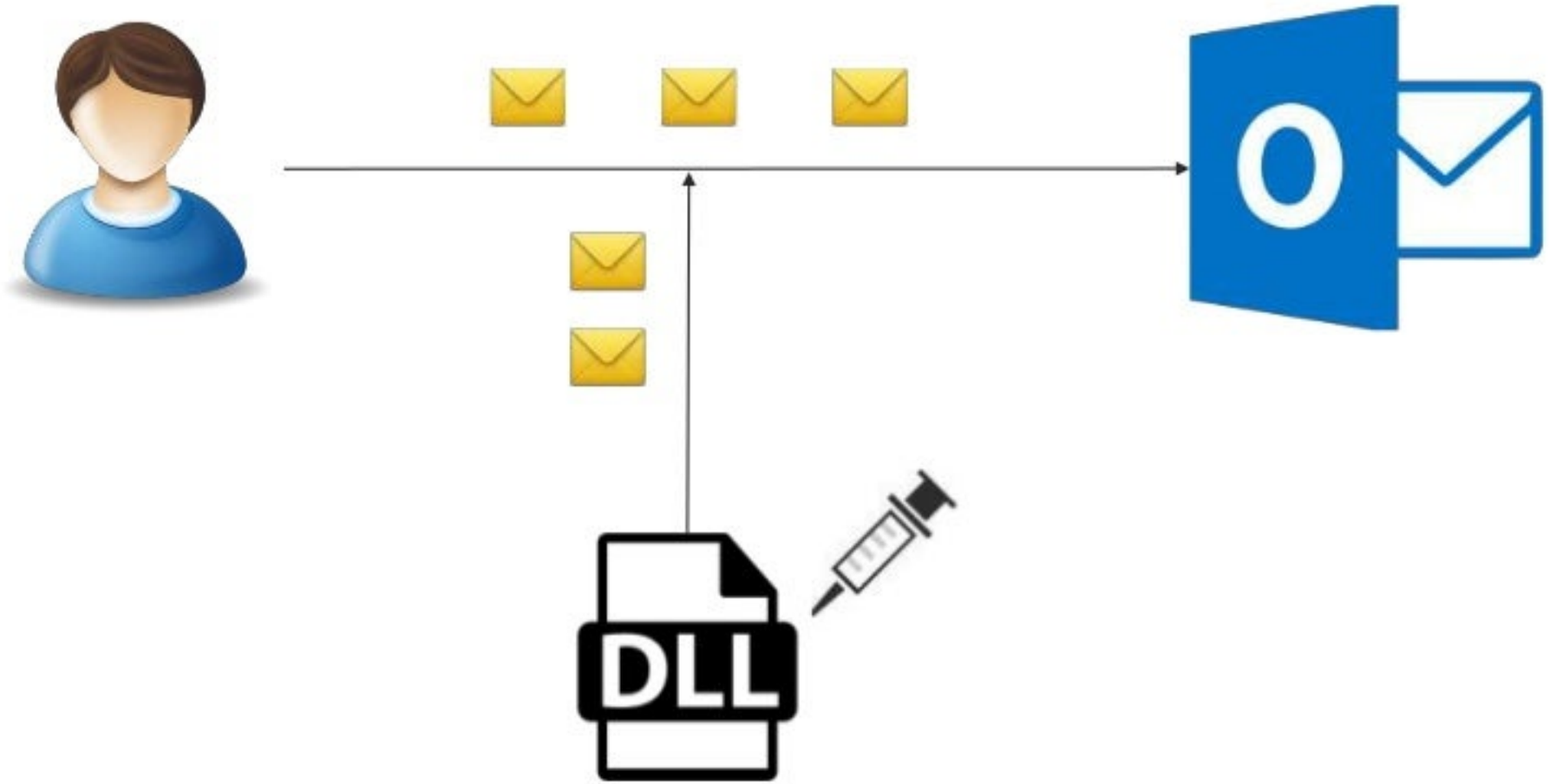
Introduction to hooks / **Scope of applications**

solarwinds msp

- Loggers
- Debuggers
- Monitoring systems
- Antivirus
- Statistics
- Analyzers

- Adware
- Spyware
- Keylogger
- Rootkit
- Trojan

Introduction to hooks / **Helpful application example**



Introduction to hooks / **Pros and cons**

solarwinds msp

⊕

- Not always require administrator rights and increased integration level
- Easy to implement
- Do not require digital sign

⊖

- Only in one process
- Require additional work to inject dll into the target process
- Big risk of being detected

Dll injection methods

solarwinds msp

Dll injection methods / **Use Windows regedit**

Registry Editor

File Edit View Favorites Help

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows

Sensor
SoftwareProtectionPlatf
SPP
SRUM
Superfetch
SvcHost
SystemRestore
Terminal Server
Time Zones
TokenBroker
Tracing
UAC
UnattendedSettings
UnattendSettings
Update
Userinstallable.drivers
VersionsList
Virtualization
VolatileNotifications
WbemPerf
WiFiDirectAPI
Windows
Winlogon
WinSAT
WinSATAPI
WirelessDocking
WOF
WUDF
Windows Performance Toolkit
Windows Phone

Name	Type	Data
(Default)	REG_SZ	mnmsrvc
AppInit_DLLs	REG_SZ	
DdeSendTimeout	REG_DWORD	0x00000000 (0)
DesktopHeapLogging	REG_DWORD	0x00000001 (1)
DeviceNotSelectedTimeout	REG_SZ	15
DwmInputUseIoCompletionPort	REG_DWORD	0x00000001 (1)
EnableDwmInputProcessing	REG_DWORD	0x00000007 (7)
EnableMitInputProcessing	REG_DWORD	0x00000007 (7)
GDIProcessHandleQuota	REG_DWORD	0x00002710 (10000)
IconServiceLib	REG_SZ	IconCodecService.dll
LoadAppInit_DLLs	REG_DWORD	0x00000000 (0)
NaturalInputHandler	REG_SZ	Ninput.dll
ShutdownWarningDialogTimeout	REG_DWORD	0xffffffff (4294967295)
Spooler	REG_SZ	yes
ThreadUnresponsiveLogTimeout	REG_DWORD	0x00000f4 (500)
TransmissionRetryTimeout	REG_SZ	90
USERNestedWindowLimit	REG_DWORD	0x00000032 (50)
USERPostMessageLimit	REG_DWORD	0x00002710 (10000)
USERProcessHandleQuota	REG_DWORD	0x00002710 (10000)
Win32kLastWriteTime	REG_SZ	1D33928A8E92551

AppInit_DLLs

LoadAppInit_DLLs

Dll injection methods / **Use Windows Events Hooks**

Dll injection methods / **Use Windows Events Hooks**

solarwinds msp

```
LRESULT CALLBACK hkproc(int code, WPARAM wparam, LPARAM lparam)
{
    return CallNextHookEx(nullptr, code, wparam, lparam);
}
```

```
std::wstring path = L"/Path/to/hook.dll";
HMODULE module = LoadLibraryW(path.c_str());
if (module)
{
    HOOKPROC proc = reinterpret_cast<HOOKPROC>(::GetProcAddress(module, "hkproc"));
    if (proc)
    {
        HHOOK hook = ::SetWindowsHookEx(
            WH_MOUSE,
            proc,
            module,
            0); // dwThreadId
    }
}
```

Dll injection methods / **Use threads**

solarwinds msp

Basic steps

- Allocate memory in target process
- Prepare opcode to load Library
- Write prepared code to target process

Set Thread context

- Use existing thread to launch injected code

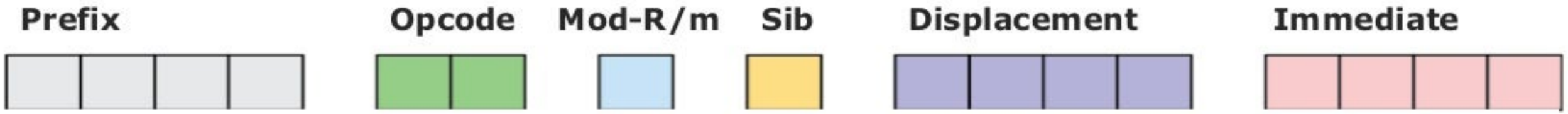
Create remote Thread

- Start new thread with injected code

Splicing hook

Splicing hook / **Machine code instructions**

solarwinds msp

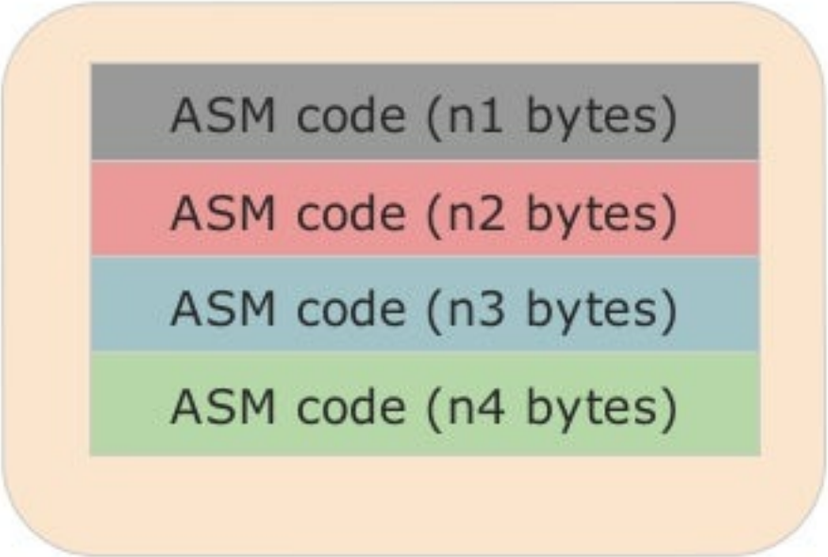


Prefix	**F0 8b D1**	:	**LOCK MOV EDX, ECX**
Opcode	**90**	:	**NOP**
Mod-R/m	**8B D1**	:	**MOV EDX, ECX**
Sib	**8B 04 31**	:	**MOV EAX, DWORD [ESI+ECX]** (SIB says that ESI is moved by ECX in this instruction)
Displacement	**8B 84 31 FE CA 00 00**	:	**MOV EAX, DWORD [ESI+ECX+CAFE]** (moved with -0x10 displacement)
Immediate	**C7 04 8F BE BA FE CA**	:	**MOV [EDI+ECX*4], 0xCAFEBABE** (0xCAFEBABE is a 4-byte immediate)

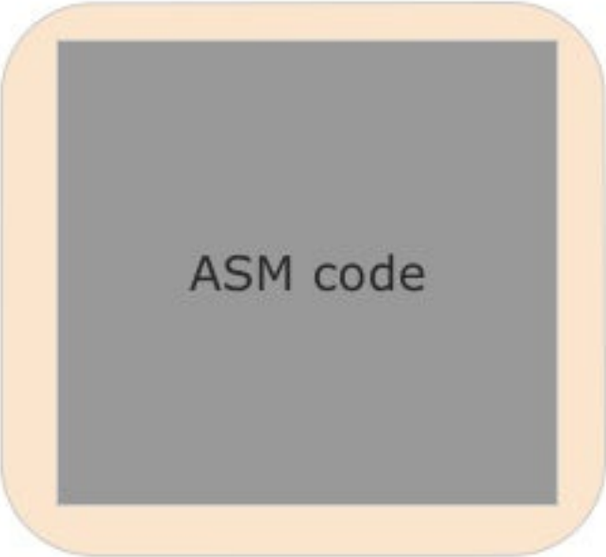
Splicing hook / **Basic hook scheme**

solarwinds msp

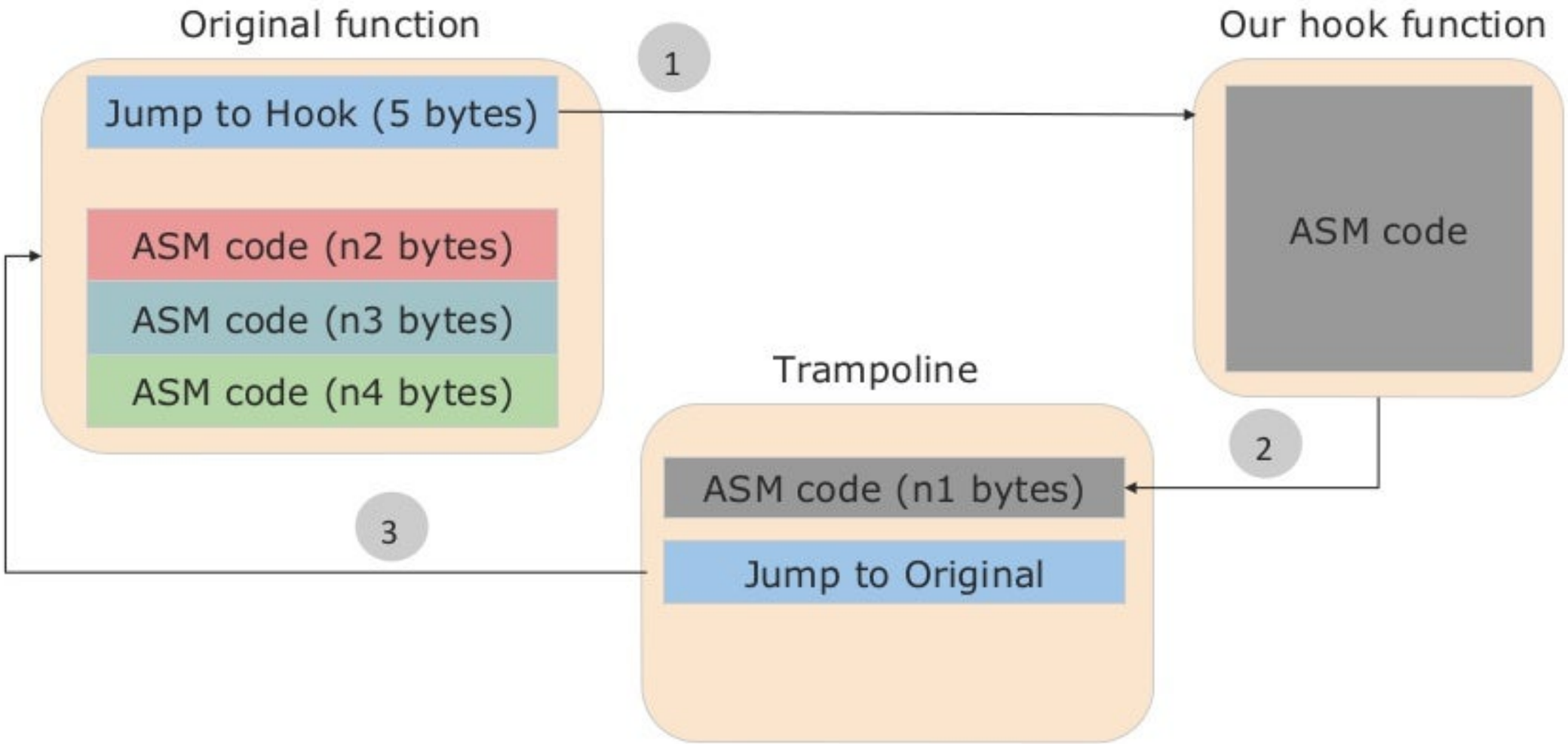
Original function



Our hook function



Splicing hook / **Basic hook scheme**



solarwinds msp

Why 5 bytes ?

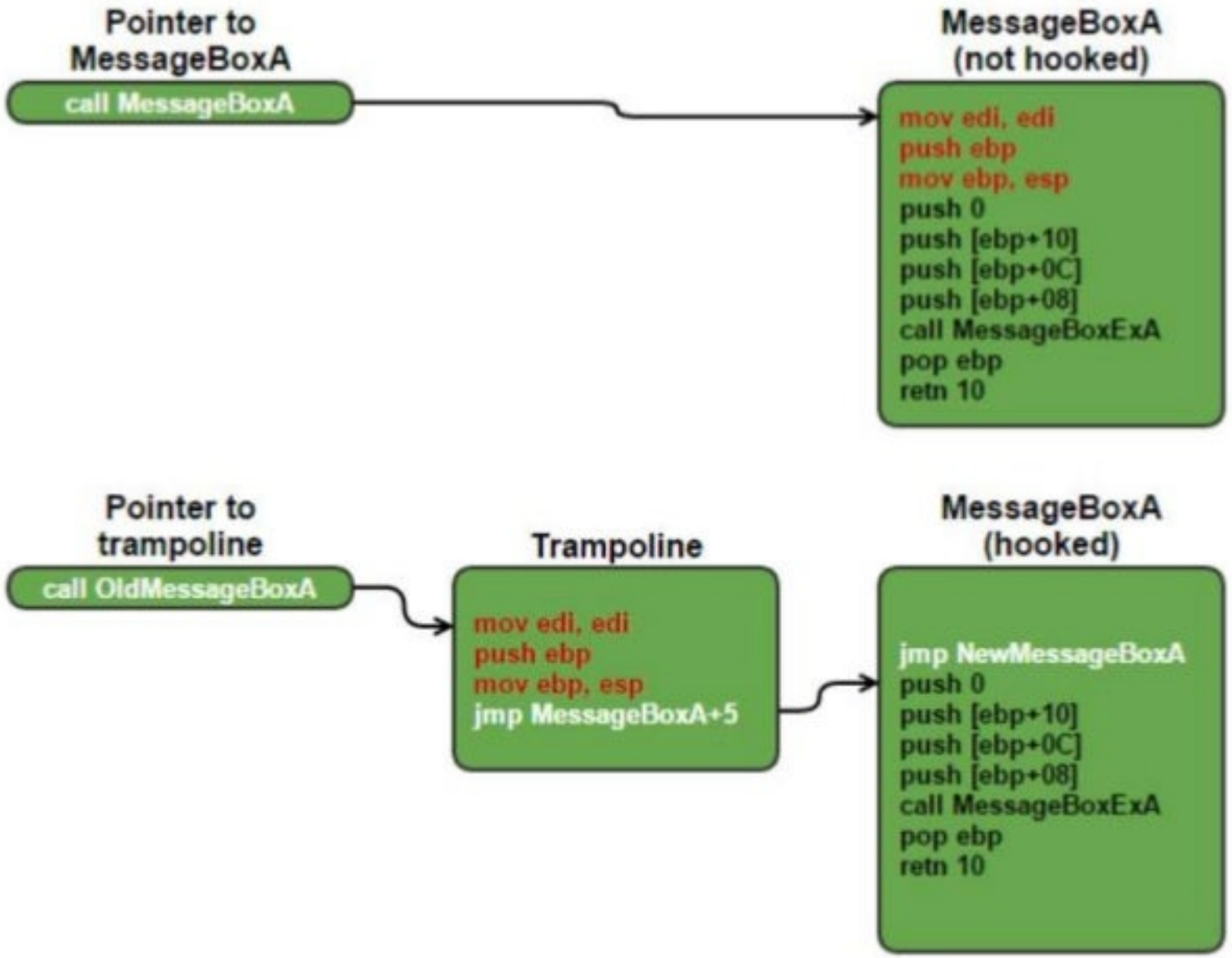
E9 2B1D778b -> hook Jump (5 bytes)

E9 -> Opcode

2B1D778b -> jump address

jump address = ToAddress - FromAddress

Splicing hook / **Basic hook scheme real example**



Splicing hook / **Set hook**

```
7554F8B0 MessageBox $  8BFF          MOV EDI,EDI
7554F8B2            .  55            PUSH EBP
7554F8B3            .  8BEC          MOV EBP,ESP
7554F8B5            .  6A FF         PUSH -1
7554F8B7            .  6A 00         PUSH 0
7554F8B9            .  FF75 14       PUSH DWORD PTR SS:[ARG.4]
7554F8BC            .  FF75 10       PUSH DWORD PTR SS:[ARG.3]
7554F8BF            .  FF75 0C       PUSH DWORD PTR SS:[ARG.2]
7554F8C2            .  FF75 08       PUSH DWORD PTR SS:[ARG.1]
7554F8C5            .  E8 D6010000   CALL MessageBoxTimeoutA
7554F8CA            .  5D            POP EBP
7554F8CB            .  C2 1000       RETN 10
```

Hook

```
7554F8B0 $  E9 2B1D778B   JMP HookMessageBoxA
7554F8B5 .  6A FF         PUSH -1
7554F8B7 .  6A 00         PUSH 0
```

Trampoline

Call original fun

```
7566002C    8BFF        MOV EDI,EDI
7566002E    55          PUSH EBP
7566002F    8BEC        MOV EBP,ESP
75660031 ^  E9 7FF8EEFF JMP 7554F8B5
```

```
int WINAPI HookMessageBoxA(
    _In_opt_ HWND hWnd,
    _In_opt_ LPCTSTR lpText,
    _In_opt_ LPCTSTR lpCaption,
    _In_ UINT uType)
{
    return OriginalMessageBoxA(hWnd, lpText,
                               lpCaption, uType);
}
```

solarwinds msp

Instructions not equal 5 bytes length ?

8B 8C 72 00 00 00 77 | **MOV ECX,DWORD PTR DS:[ESI*2+EDX+77000000]**

Solution: use disassembler library

8B 8C 72 00 00 00 77 → disassembler → Asm code

MOV ECX,DWORD PTR DS:[ESI*2+EDX+77000000]

Splicing hook / **Jump instructions handling**

solarwinds msp

```
008263B6  ·  8B0D 58F38200   MOV ECX,DWORD PTR DS:[argv]
008263BC  ·  51              PUSH ECX
008263BD  ·  8B15 54F38200   MOV EDX,DWORD PTR DS:[argc]
008263C3  ·  52              PUSH EDX
008263C4  ·  E8 5AACFFFF     CALL 00821023                    [wmain
008263C9  ·  83C4 0C         ADD ESP,0C
```

```
00821019  ·  E9 7A780000   JMP ?_Orphan_all@_Container_base12@std@@QAEXXZ
0082101E  ·  E9 6D270000   JMP std::_Generic_error_category::`scalar deletin
00821023  $  E9 F8450000   JMP wmain
00821028  ·  E9 A3410000   JMP std::_Iostream_error_category::message
0082102D  ·  E9 BE630000   JMP _RTC_GetErrDesc
00821032  $  E9 391A0000   JMP std::allocator<char>::construct<char *,char *
```

Splicing hook / **Injecting into WinApi functions (Hot patch)**

```
74735F9B  .  CC        INT3
74735F9C  .  CC        INT3
74735F9D  .  CC        INT3
74735F9E  .  CC        INT3
74735F9F  .  CC        INT3
74735FA0  $  8BFF      MOV EDI,EDI
74735FA2  .  55        PUSH EBP
74735FA3  .  8BEC      MOV EBP,ESP
```

WinApi Hot Patch

```
74735F9B   E9 10000000   JMP 74735FB0
74735FA0   EB F9         JMP SHORT 74735F9B
74735FA2   55            PUSH EBP
74735FA3   8BEC          MOV EBP,ESP
```

Short relative **Jump**

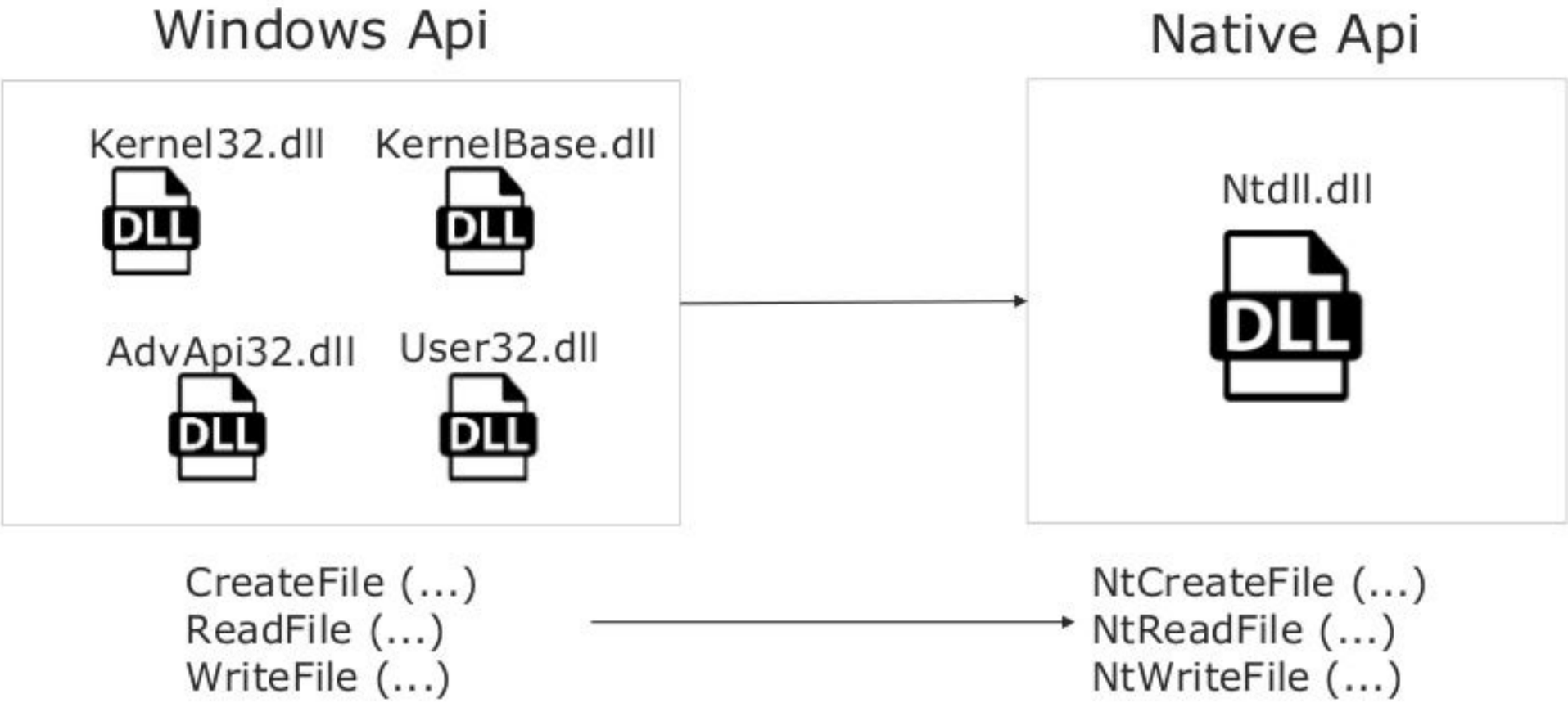
ASM: JMP relative offset **0x00-0x7F** and Reverse **0x80-0xFF**

Opcode: EB offset

Protection from hooking

solarwinds msp

Protection from hooking / **Windows Api levels**



Protection from hooking / **System call instructions**

solarwinds msp

- Int 2E
- Sysenter (x86)
- Syscall (x64)

CPU Model Specific Register (MSR) SYSENTER_EIP_MSR

Protection from hooking / **Sysenter**

solarwinds msp

```
__declspec(naked) void FastSystemCall( )
{
    __asm
    {
        mov edx, esp;
        // SYSENTER
        _emit 0x0F;
        _emit 0x34;
    }
}
```

```
EB 03    JMP SHORT KiFastSystemCall
CC       INT3
CC       INT3
CC       INT3
8BD4     MOV EDX,ESP
0F34     SYSENTER
```

```
__declspec(naked) NTSTATUS WINAPI SysenterZwReadFile(...)
{
    __asm
    {
        mov EAX, 0xB7
        call FastSystemCall
        retn 24
    }
}
```

Protection from hooking / **Pros and cons**

solarwinds msp

⊕

- The most effective method
- Allows to intercept absolutely any function in process
- Code tuning (for example obfuscation)
- Low risk of being detected
- Cross-platform
- Can be set on top of other

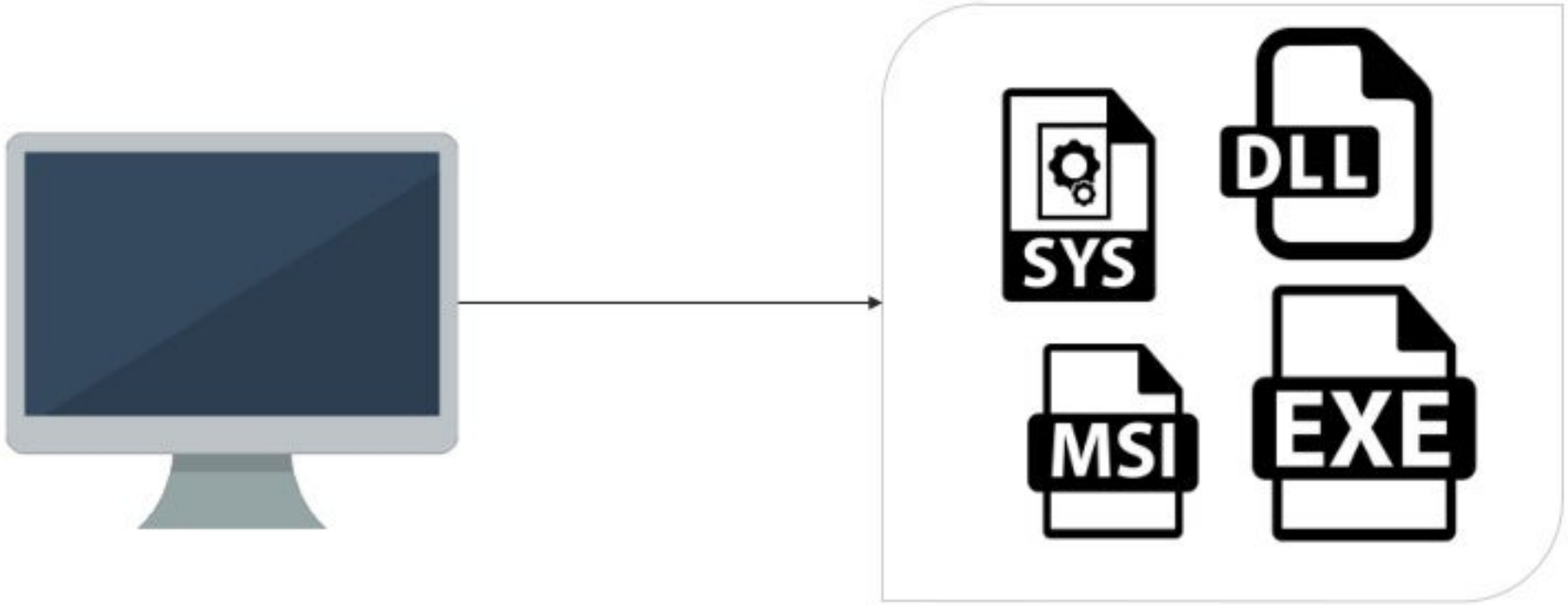
⊖

- Thread safe problems
- Target dll might not presented at the same time
- Direct memory page modification

IAT/EAT hooks

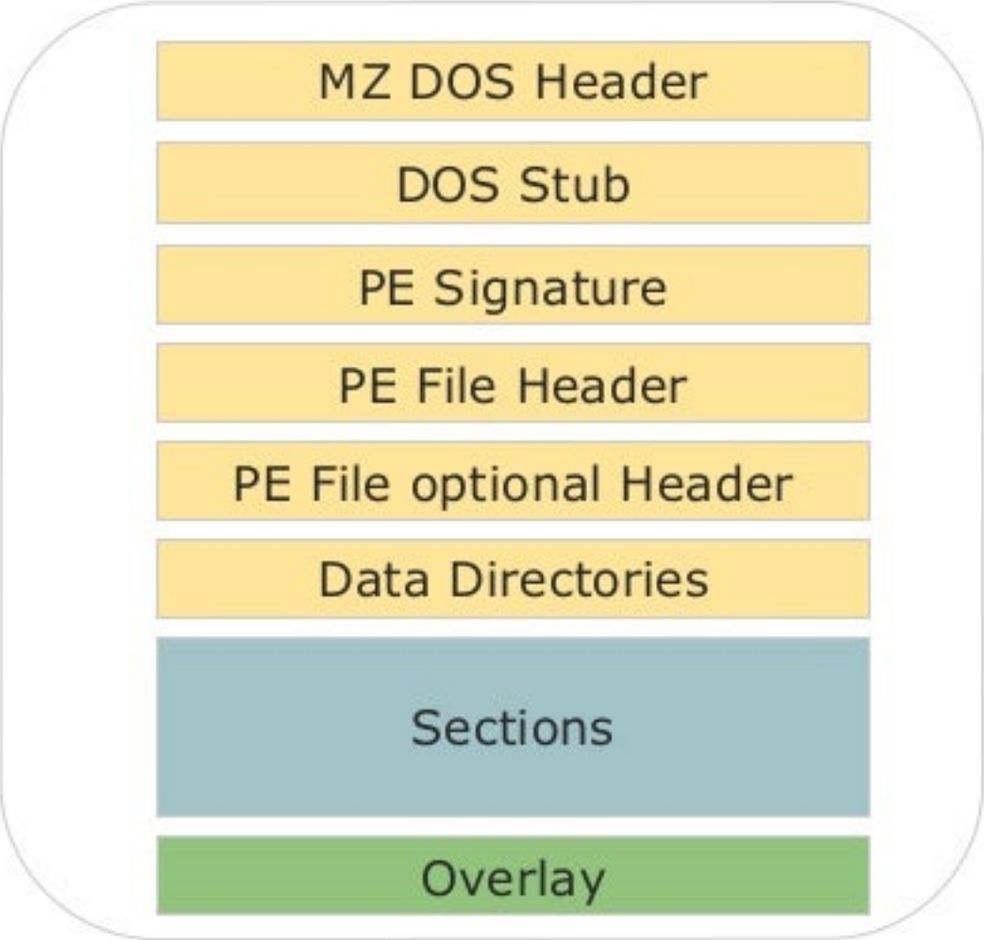
```
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 e8 00 00 00  |................|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |........!..L.!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  52 85 94 94 16 e4 fa c7  16 e4 fa c7 16 e4 fa c7  |R...............|
00000090  0d 79 51 c7 0f e4 fa c7  0d 79 64 c7 18 e4 fa c7  |.yQ......yd.....|
000000a0  0d 79 50 c7 73 e4 fa c7  1f 9c 69 c7 11 e4 fa c7  |.yP.s.....i.....|
000000b0  16 e4 fb c7 4c e4 fa c7  08 b6 7e c7 17 e4 fa c7  |....L.....~.....|
000000c0  08 b6 6e c7 17 e4 fa c7  08 b6 6b c7 17 e4 fa c7  |..n.......k.....|
000000d0  52 69 63 68 16 e4 fa c7  00 00 00 00 00 00 00 00  |Rich............|
000000e0  00 00 00 00 00 00 00 00  50 45 00 00 4c 01 05 00  |........PE..L...|
000000f0  c4 df 90 51 00 00 00 00  00 00 00 00 e0 00 02 01  |...Q............|
00000100  0b 01 09 00 00 98 00 00  00 64 01 00 00 00 00 00  |.........d......|
00000110  67 20 00 00 00 10 00 00  00 b0 00 00 00 00 40 00  |g .............@.|
00000120  00 10 00 00 00 02 00 00  05 00 00 00 00 00 00 00  |................|
00000130  05 00 00 00 00 00 00 00  00 50 02 00 00 04 00 00  |.........P......|
00000140  00 00 00 00 02 00 00 80  00 00 10 00 00 10 00 00  |................|
00000150  00 00 10 00 00 10 00 00  00 00 00 00 10 00 00 00  |................|
00000160  00 00 00 00 00 00 00 00  cc ce 00 00 50 00 00 00  |............P...|
00000170  00 50 01 00 7c e2 00 00  00 00 00 00 00 00 00 00  |.P..|...........|
00000180  00 00 00 00 00 00 00 00  00 40 02 00 3c 08 00 00  |.........@..<...|
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

IAT/EAT hooks / **Portable Executable**



IAT/EAT hooks / **Portable Executable (format)**

solarwinds msp

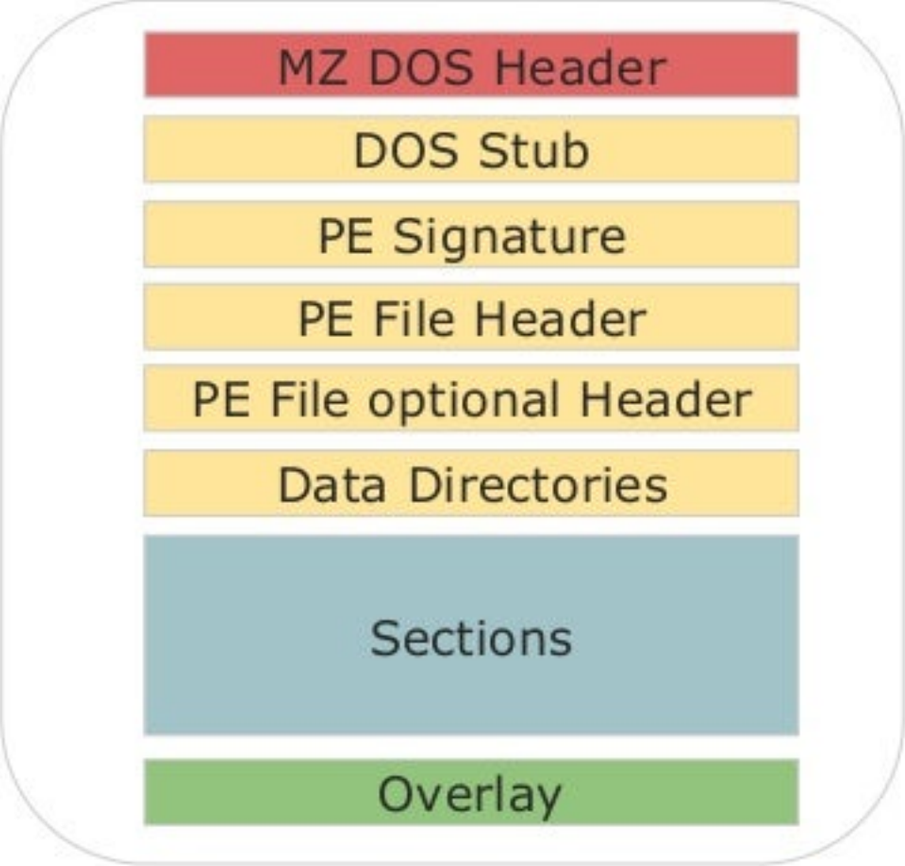


IAT/EAT hooks / **Portable Executable (MZ DOS Header)**

solarwinds msp

```
00000000h: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 ; MZ ...
```

```
typedef struct _IMAGE_DOS_HEADER {      // DOS .EXE header
    WORD   e_magic;                     // Magic number
    WORD   e_cblp;                      // Bytes on last page of file
    WORD   e_cp;                        // Pages in file
    WORD   e_crlc;                      // Relocations
    WORD   e_cparhdr;                   // Size of header in paragraphs
    WORD   e_minalloc;                  // Minimum extra paragraphs needed
    WORD   e_maxalloc;                  // Maximum extra paragraphs needed
    WORD   e_ss;                        // Initial (relative) SS value
    WORD   e_sp;                        // Initial SP value
    WORD   e_csum;                      // Checksum
    WORD   e_ip;                        // Initial IP value
    WORD   e_cs;                        // Initial (relative) CS value
    WORD   e_lfarlc;                    // File address of relocation table
    WORD   e_ovno;                      // Overlay number
    WORD   e_res[4];                    // Reserved words
    WORD   e_oemid;                     // OEM identifier (for e_oeminfo)
    WORD   e_oeminfo;                   // OEM information; e_oemid specific
    WORD   e_res2[10];                  // Reserved words
    LONG   e_lfanew;                    // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```



IAT/EAT hooks / **Portable Executable (Sections)**

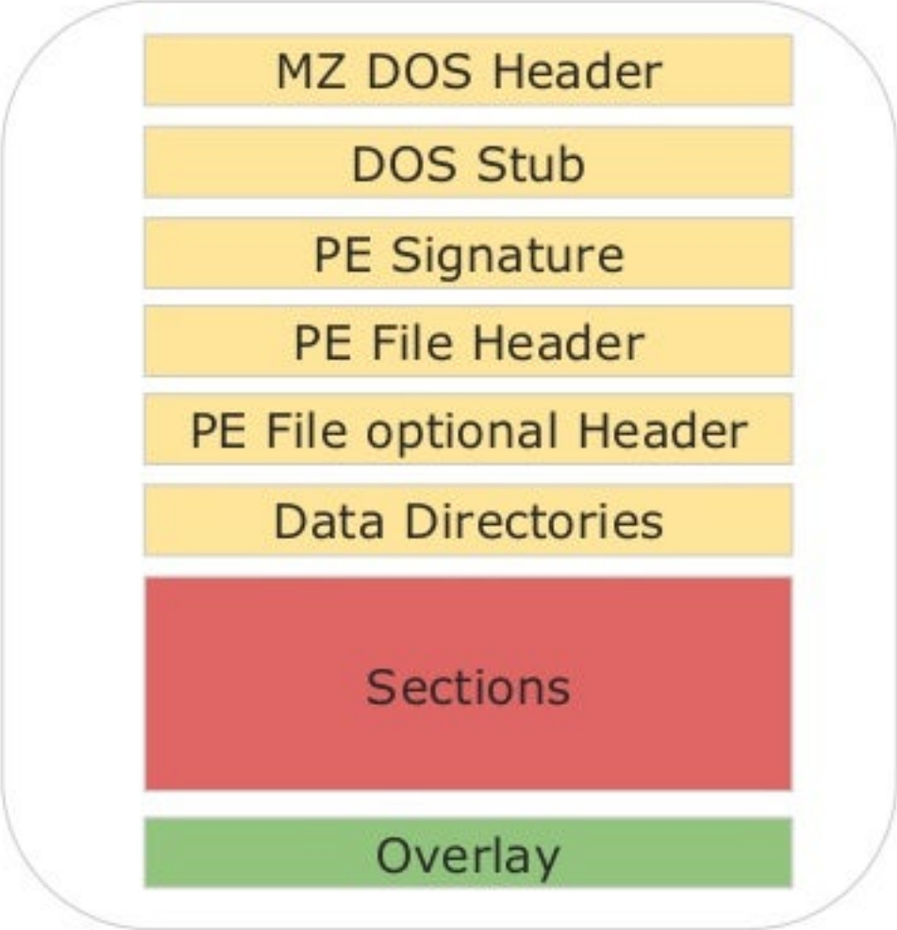
solarwinds msp

.text
.rdata
.data
.idata
.rsrc
.mySection

Create own section in PE

```
#pragma section("mySection", read, write)

__declspec(allocate("mySection")) int value = 0;
```



solarwinds msp

- RVA (Relative virtual address)
- VA (Virtual address)

VA = Imagebase + RVA

Imagebase - address where module was mapped

Example:

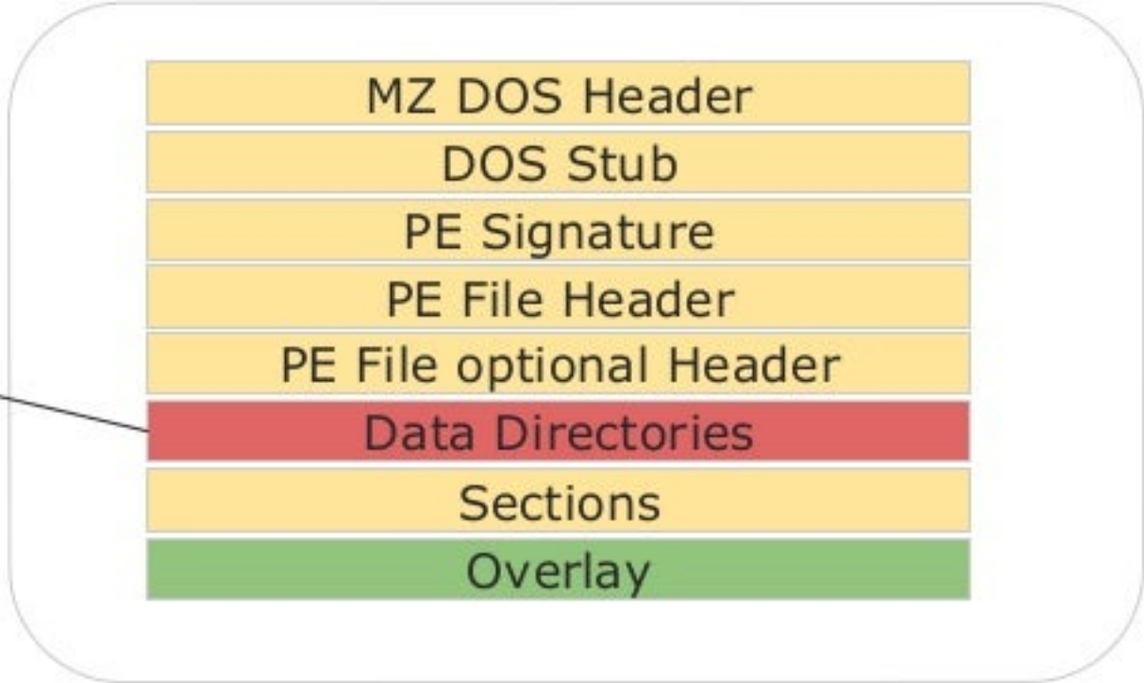
module = 0x00ee0000

pDosHeader->e_lfanew is 0x000000f0 (RVA)

VA = module + pDosHeader->e_lfanew (0x00ee00f0)

IAT/EAT hooks / **PE Import table**

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;
        DWORD   OriginalFirstThunk;
    } DUMMYUNIONNAME;
    DWORD   TimeDateStamp;
    DWORD   ForwarderChain;
    DWORD   Name;
    DWORD   FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;
```



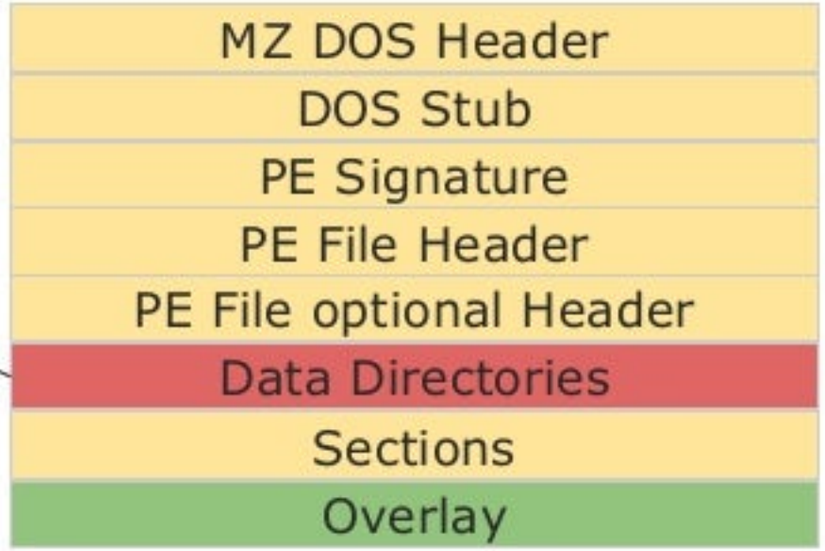

```
HMODULE hModule = GetModuleHandle(NULL);
PIMAGE_DOS_HEADER pDosHeader = PIMAGE_DOS_HEADER(hModule);
PIMAGE_NT_HEADERS pNTHeaders =
    (PIMAGE_NT_HEADERS)RvaToVa(hModule, pDosHeader->e_lfanew);

PIMAGE_DATA_DIRECTORY pImport =
    (PIMAGE_DATA_DIRECTORY)&pNTHeaders->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_IMPORT];
```

IAT/EAT hooks / **PE Export table**

solarwinds msp

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;
    DWORD   Base;
    DWORD   NumberOfFunctions;
    DWORD   NumberOfNames;
    DWORD   AddressOfFunctions;
    DWORD   AddressOfNames;
    DWORD   AddressOfNameOrdinals;
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```




```
PIMAGE_DATA_DIRECTORY pExportDirectoryData =
    &pNTHeaders->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXPORT];

PIMAGE_EXPORT_DIRECTORY pExportDirectory =
    (PIMAGE_EXPORT_DIRECTORY)RvaToVa(hModule, pExportDirectoryData->VirtualAddress);
```

IAT/EAT hooks / **Real PE Import Table**

solarwinds msp

00980000	00001000	Application		PE header
00981000	00001000	Application	.text	Code
00982000	00001000	Application	.rdata	Imports
00983000	00001000	Application	.data	Data
00984000	00001000	Application	.rsrc	Resources

RVA	Name
00613C02h	KERNEL32.dll
00613C30h	USER32.dll
00613D7Ah	ADVAPI32.dll
00613D9Eh	SHELL32.dll
00613DF2h	ole32.dll
00613DFCh	OLEAUT32.dll
00613E46h	WTSAPI32.dll
00613EC4h	WS2_32.dll
00613EDEh	RPCRT4.dll
00613F14h	WLDAP32.dll
00613FFAh	VERSION.dll
0061403Ah	USERENV.dll
006140BCh	SHLWAPI.dll

RVA	Hint	Name
005A9320h	0033h	wsprintfW
005A9324h	007Eh	GetSystemMetrics
005A9328h	008Bh	GetUserObjectInformationW
005A932Ch	0068h	GetProcessWindowStation
005A9330h	0023h	GetDesktopWindow
005A9334h	000Eh	MessageBoxA

PE Import Table hook

solarwinds msp

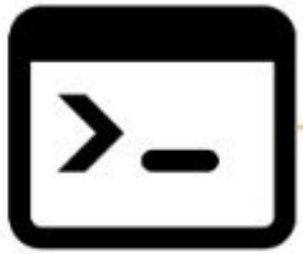
PE Import Table hook / **PE Import table**

solarwinds msp

User32.dll
.text section
MessageBoxA

Import table for User32.dll

Address	Hex dump	ASCII (ANSI - Cy
00982040	10 0E 1E 77 50 57 73 74 A0 72 73 74 60 66 73 74	▯▯ wPWst rst`fst
00982050	30 87 73 74 00 00 00 00 90 BD C8 69 30 47 CB 69	0‡st ħSИi0GЛi
00982060	34 47 CB 69 1B B7 C1 69 04 CC C8 69 0A 38 C8 69	4GЛi▯·Бi▯МИi 8Иi
00982070	34 09 C1 69 6F 26 C1 69 10 09 C1 69 E6 A9 C1 69	4 Бio&Бi▯ Бiж©Бi
00982080	0C CC C8 69 2C AF C8 69 B5 C3 C1 69 45 26 C1 69	▯МИi,ЇИiµГБiE&Бi
00982090	27 26 C1 69 14 47 CB 69 0C 7B C2 69 A1 B8 C8 69	'&Бi▯GЛi▯{ВiЎёИi
009820A0	36 BF C5 69 22 7B C2 69 D2 D1 C1 69 5B BF C5 69	6їEi"{ВiТСБi[їEi
009820B0	51 60 C6 69 87 BC C5 69 80 26 C0 69 00 00 00 00	Q`Жi‡jEiЂ&Ai
009820C0	B0 F8 54 75 00 00 00 00 00 00 00 00 7E 12 98 00	°шTu ~▯



MessageBoxA

mov edi,dword ptr [__imp__MessageBoxA (9820C0h)]
…
call edi

PE Import Table hook / **PE Import table hook**

solarwinds msp

Import table for User32.dll

Address	Hex dump	ASCII (ANSI - Cy
00982040	10 0E 1E 77 50 57 73 74 A0 72 73 74 60 66 73 74	▯▯ wPWst rst`fst
00982050	30 87 73 74 00 00 00 00 90 BD C8 69 30 47 CB 69	0‡st ħSИi0GЛi
00982060	34 47 CB 69 1B B7 C1 69 04 CC C8 69 0A 38 C8 69	4GЛi▯·Бi▯МИi 8Иi
00982070	34 09 C1 69 6F 26 C1 69 10 09 C1 69 E6 A9 C1 69	4 Бio&Бi▯ Бiж®Бi
00982080	0C CC C8 69 2C AF C8 69 B5 C3 C1 69 45 26 C1 69	▯МИi,ЇИiµГБiE&Бi
00982090	27 26 C1 69 14 47 CB 69 0C 7B C2 69 A1 B8 C8 69	'&Бi▯GЛi▯{ВiЎёИi
009820A0	36 BF C5 69 22 7B C2 69 D2 D1 C1 69 5B BF C5 69	6ЇEi"{ВiТСБi[ЇEi
009820B0	51 60 C6 69 87 BC C5 69 80 26 C0 69 00 00 00 00	Q`Жi‡jEiЂ&Ai
009820C0	~~B0 F8 54 75~~ 00 00 00 00 00 00 00 00 7E 12 98 00	°шTu ~▯

70 11 98 00

```
PIMAGE_IMPORT_DESCRIPTOR pImportTable =
    (PIMAGE_IMPORT_DESCRIPTOR)RvaToVa(hModule, pDirectoryData->VirtualAddress);
PIMAGE_THUNK_DATA pNamesTable = (PIMAGE_THUNK_DATA)RvaToVa(hModule, pImportTable->OriginalFirstThunk);
LPVOID *targetFunction =
    (LPVOID*)RvaToVa(pNamesTable, pImportTable->FirstThunk - pImportTable->OriginalFirstThunk);

::InterlockedExchange((PLONG)targetFunction, *(PLONG)&MyMessageBox);
```

PE Export Table hook

PE Export Table hook / **PE Export table**

solarwinds msp

User32.dll
.text section
MessageBoxA

Export table for User32.dll

754E1000	00081000	USER32	.text	Code,exports
	7555B3F4	·	D0980100	
	7555B3F8	·	40EF0100	
	7555B3FC	·	C0B20600	
	7555B400	·	40B30600	
	7555B404	·	10420700	
	7555B408	·	B0F80600	
	7555B40C	·	E0F80600	
	7555B410	·	10F90600	

```
typedef int (WINAPI *Type)(HWND, LPCTSTR, LPCTSTR, UINT);

Type originalFunction = (Type)GetProcAddress(GetModuleHandle(L"User32.dll"), "MessageBoxA");

originalFunction(0, NULL, NULL, MB_OK);
```

PE Export Table hook / **PE Export table hook**

solarwinds msp

User32.dll
.text section
MessageBoxA

Export table for User32.dll

754E1000	00081000	USER32	.text	Code,exports
	7555B3F4	·	D0980100	
	7555B3F8	·	40EF0100	
	7555B3FC	·	C0B20600	
	7555B400	·	40B30600	
	7555B404	·	10420700	
	7555B408	·	30123E8B	
	7555B40C	·	E0F80600	
	7555B410	·	10F90600	

```
typedef int (WINAPI *Type)(HWND, LPCTSTR, LPCTSTR, UINT);

Type hookFunction = (Type)GetProcAddress(GetModuleHandle(L"User32.dll"), "MessageBoxA");

hookFunction(0, NULL, NULL, MB_OK);
```

PE Export Table hook / **PE IT/ET hooks vs Splicing**

solarwinds msp

⊕

- Easy to implement
- There is no memory modification
- Target dll already presented
- Do not require code maintenance
- Thread-safe

⊖

- Big risk of being detected
- May be replaced by other hook

C++ virtual table

solarwinds msp

C++ virtual table / **C++ virtual table**

```
class Task
{
public:
	Task() {}

	virtual ~Task() {}

	virtual int Process(int a, int b) = 0;
};

class TaskImpl : public Task
{
public:
	TaskImpl() {}

	virtual ~TaskImpl() {}

	virtual int Process(int a, int b)
	{
		return a + b;
	}
};
```

Task class vTable

Task::~Task()
Task::Process()

TaskImpl class vTable

TaskImpl::~TaskImpl()
TaskImpl::Process()

C++ virtual table / **Calling conventions (Microsoft compilers)**

solarwinds msp

32 bit architecture

thiscall: this -> ECX/RCX. Params through Stack

Fastcall: the first and second params through ECX, EDX, other through stack

stdcall: all params through stack

64 bit architecture

Fastcall: from 1 to 4 param through RCX, RDX, R8, R9, other through stack

C++ virtual table / **Virtual Table (Debug configuration)**

solarwinds msp

```
Task* task = new TaskImpl();
```

1

```
00961B4F  call   operator new (09612C1h)
00961B57  mov  dword ptr [ebp-0ECh],eax
00851B6D  mov  ecx,dword ptr [ebp-0ECh]
00851B73  call   TaskImpl::TaskImpl (08510FAh)
```

2

```
008518B0  mov dword ptr [this],ecx
008518B6  call  Task::Task (0851163h)
008518BB  mov eax,dword ptr [this]
008518BE  mov dword ptr [eax],offset TaskImpl::`vftable' (0858B44h)
```

3

```
00851860  mov  dword ptr [this],ecx
00851863  mov  eax,dword ptr [this]
00851866  mov  dword ptr [eax],offset Task::`vftable' (0858B34h)
```

4

```
task->Process(1, 2);
```

```
00961BA8  push 2
00961BAA  push 1
00961BAC  mov   eax,dword ptr [task]
00961BAF  mov  edx,dword ptr [eax]
00961BB1  mov  ecx,dword ptr [task]
00961BB4  mov  eax,dword ptr [edx+4]
00961BB7  call   eax
```

C++ virtual table / **Virtual Table (Release x64 configuration)**

solarwinds msp

```
Task* task = new TaskImpl();
task->Process(1, 2);
```

```
00007FF7F6B71059  call   operator new (07FF7F6B710CCh)
00007FF7F6B7105E  mov    edx,1
00007FF7F6B71063  mov    qword ptr [rsp+30h],rax
00007FF7F6B71068  lea    rcx,[TaskImpl::`vftable' (07FF7F6B732E0h)]
00007FF7F6B7106F  mov    qword ptr [rax],rcx
00007FF7F6B71072  mov    rcx,rax
00007FF7F6B71075  lea    r8d,[rdx+1]
00007FF7F6B71079  call   qword ptr [TaskImpl::`vftable'+8h (07FF7F6B732E8h)]
```

vTable memory

Address: 0x00007FF7F6B732E8

```
0x00007FF7F6B732E8  30 10 b7 f6 f7 7f 00 00 e0 36 b7 f6 f7 7f 00 00
0x00007FF7F6B73313  00 ce a0 c0 5a 00 00 00 00 02 00 00 00 84 00 00
```

C++ virtual table hook

solarwinds msp

C++ virtual table hook / **Virtual Table Hook algorithm**

```
void* vTableHook(void* instance, void* hook, int offset)
{
        DWORD_PTR vtable =    *(DWORD_PTR*)instance;
        DWORD_PTR entry =     (DWORD_PTR)vtable + sizeof(DWORD_PTR) * offset;
        DWORD_PTR original =  *((DWORD_PTR*)entry);

        DWORD dwOldProtect = 0;
        ::VirtualProtect((void*)entry, sizeof(DWORD_PTR),
              PAGE_EXECUTE_READWRITE, &dwOldProtect);

#ifdef _WIN64
        ::InterlockedExchange64((PLONGLONG)entry, (LONGLONG)hook);
#else
        ::InterlockedExchange((PLONG)entry, (LONG)hook);
#endif

        ::VirtualProtect((void*)entry, sizeof(DWORD_PTR), dwOldProtect, &dwOldProtect);

        return (void*)original;
}
```

C++ virtual table hook / **Virtual Table Hook (separated function)**

solarwinds msp

```
typedef int (__thiscall *FunctionType)(Task*, int, int);

FunctionType OriginalFunction = 0;

int __fastcall ProcessHook(Task* _this, void* edx, int a, int b)
{
    return OriginalFunction(_this, a, b);
}

int main()
{
    Task* task = new TaskImpl();

    OriginalFunction = (FunctionType)vTableHook(task, &ProcessHook, 1);
    task->Process(1, 2);

    return 0;
}
```

Task class vTable

0: Task::~Task()
1: Task::Process()

C++ Microsoft Component Object Model (COM) virtual table hook

solarwinds msp

Microsoft Com / **COM Virtual Table**

solarwinds msp

```
MIDL_INTERFACE("00000000-0000-0000-C000-000000000046")
IUnknown
{
public:
        virtual HRESULT __stdcall QueryInterface(REFIID riid, void** ppvObject) = 0;
        virtual ULONG __stdcall AddRef() = 0;
        virtual ULONG __stdcall Release() = 0;
}
```

```
MIDL_INTERFACE("00020400-0000-0000-C000-000000000046")
IDispatch : public IUnknown
{
public:
        virtual HRESULT __stdcall GetTypeInfoCount(...) = 0;
        virtual HRESULT __stdcall GetTypeInfo(...) = 0;
        virtual HRESULT __stdcall GetIDsOfNames(...) = 0;
        virtual HRESULT __stdcall Invoke(...) = 0;
};
```

```
MIDL_INTERFACE("A6EF9860-C720-11d0-9337-00A0C90DCAA9")
IDispatchEx : public IDispatch
{
public:
        virtual HRESULT __stdcall GetDispID(...) = 0;

        virtual HRESULT __stdcall InvokeEx(...) = 0;
};
```

Microsoft Com / **COM Virtual Table Hook**

solarwinds msp

```
typedef HRESULT(__stdcall *InvokeExType)(...);

InvokeExType InvokeExOriginal = nullptr;

HRESULT __stdcall InvokeExHook(IUnknown* inctane, ...)
{
    return InvokeExOriginal(inctane, ...);
}

int main()
{
    IUnknown* instance = ...;

    InvokeExOriginal = (InvokeExType)vTableHook(instance, &InvokeExHook, 8);
    ...
}
```

Antivirus reactions

Useful links

Hook libs:

Mhook: https://github.com/martona/mhook

EasyHook: https://easyhook.github.io/

Deviare API hook: https://www.nektra.com/products/deviare-api-hook-windows/

Disasm libs:

Urmem: https://github.com/urShadow/urmem

DsmLib: https://github.com/martona/mhook/tree/master/disasm-lib

Thanks everyone

solarwinds msp