Microsoft Research Detours Package Sample

Sample

0. Default Header File

```
#include "../../../../../cppcore/Inc/cppcore.h"
#pragma comment(lib, "cppcore.lib")

#define _X86_
#include "windef.h"
#include "WinBase.h"
#include "detours.h"
#pragma comment(lib, "detours.lib")
```

---

1. Self detour function

```
void TargetFunction()
{
        core::Log_Error(TEXT("Target Function."));
}
void (*MyTargetFunction)() = TargetFunction;

void HookingFunction()
{
        core::Log_Error(TEXT("Hooking Function."));
}

int main()
{
```

```
    DetourTransactionBegin();
    DetourUpdateThread(GetCurrentThread());
    DetourSetIgnoreTooSmall(TRUE);

    DetourAttach((PVOID*)&MyTargetFunction, HookingFunction);

    DetourTransactionCommit();
    TargetFunction();
    return 0;
}
```

---

2. Interception Sleep Function

> *SetDLL.exe*

1) DoesDLLExportOrdinal1

- 64-bit → 32-bit 타겟 프로세스를 작성하거나 또는 그 반대일때, 프로세스를 DetourCreateProcessWithDllEx or DetourCreateProcessWithDlls를 작성 시 API는 임시 헬퍼 프로세스를 작성해야 한다.
- 이때 rundll32.exe 메커니즘을 사용하여 사용자 제공 DLL의 복사본을 도우미 프로세스에로드합니다.
  - rundll32.exe이 DLL의 내보내기 된 Ordinal 1번 함수를 호출합니다.
  - DetourFinishHelperProcess API는 DLL의 Ordinal 1 함수를 export 함수로 설정해야 합니다.
    - 즉, DLL 내에 DetourFinishHelperProcess API가 Export 되었는지 확인 하는 기능
- DllMain 함수 내에서 사용자 제공 DLL은 DetourIsHelperProcess API를 호출하여 프로세스가 도우미 프로세스인지 아니면 대상 프로세스인지 확인 할 수 있습니다.
- https://documentation.help/Detours/Ove_Helpers.htm

```
D:₩SecuLetter₩git₩Detours₩bin.X86>dumpbin /exports D:₩SecuLetter₩git₩Detours₩bin.X86₩simple32.dll
Microsoft (R) COFF/PE Dumper Version 14.24.28316.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file D:₩SecuLetter₩git₩Detours₩bin.X86₩simple32.dll

File Type: DLL

  Section contains the following exports for simple32.dll

    00000000 characteristics
    FFFFFFFF time date stamp
        0.00 version
           1 ordinal base
           2 number of functions
           1 number of names

    ordinal hint RVA      name

          2    0 00001010 TimedSleepEx = ?TimedSleepEx@@YGKKH@Z (unsigned long __stdcall TimedSleepEx(unsigned long,int))
          1      00004E50 [NONAME] _DetourFinishHelperProcess@16

  Summary

        2000 .data
        2000 .detourc
        1000 .detourd
        7000 .rdata
        2000 .reloc
        1000 .rsrc
       15000 .text
```

```
static BOOL CALLBACK ExportCallback(_In_opt_ PVOID pContext,
        _In_ ULONG nOrdinal,
        _In_opt_ LPCSTR pszName,
        _In_opt_ PVOID pCode)
{
        (void)pContext;
        (void)pCode;
        (void)pszName;

        if (nOrdinal == 1) {
                *((BOOL*)pContext) = TRUE;
        }
        return TRUE;
}

BOOL DoesDllExportOrdinal1(PCHAR pszDllPath)
{
```

```
        HMODULE hDll = LoadLibraryExA(pszDllPath, NULL, DONT_R
ESOLVE_DLL_REFERENCES);
        if (hDll == NULL) {
                printf("setdll.exe: LoadLibraryEx(%s) failed w
ith error %d.\n",
                        pszDllPath,
                        GetLastError());
                return FALSE;
        }

        BOOL validFlag = FALSE;
        DetourEnumerateExports(hDll, &validFlag, ExportCallbac
k);
        FreeLibrary(hDll);
        return validFlag;
}
```

2) SetFile

> *https://wonjayk.tistory.com/268 : StringCchCopy & StringCchCat*
> *https://www.joinc.co.kr/w/man/4200/CreateFile : CreateFile*

```
static BOOL CALLBACK ListBywayCallback(_In_opt_ PVOID pContex
t,
        _In_opt_ LPCSTR pszFile,
        _Outptr_result_maybenull_ LPCSTR* ppszOutFile)
{
        (void)pContext;

        *ppszOutFile = pszFile;
        if (pszFile) {
                printf("    %s\n", pszFile);
        }
        return TRUE;
```

```
}

static BOOL CALLBACK ListFileCallback(_In_opt_ PVOID pContext,
        _In_ LPCSTR pszOrigFile,
        _In_ LPCSTR pszFile,
        _Outptr_result_maybenull_ LPCSTR* ppszOutFile)
{
        (void)pContext;

        *ppszOutFile = pszFile;
        printf("    %s -> %s\n", pszOrigFile, pszFile);
        return TRUE;
}

static BOOL CALLBACK AddBywayCallback(_In_opt_ PVOID pContext,
        _In_opt_ LPCSTR pszFile,
        _Outptr_result_maybenull_ LPCSTR* ppszOutFile)
{
        PBOOL pbAddedDll = (PBOOL)pContext;
        if (!pszFile && !*pbAddedDll) {                     //
Add new byway.
                *pbAddedDll = TRUE;
                *ppszOutFile = s_szDllPath;
        }
        return TRUE;
}

BOOL SetFile(PCHAR pszPath)
{
        BOOL bGood = TRUE;
```

```
        HANDLE hOrg = INVALID_HANDLE_VALUE;
        HANDLE hNew = INVALID_HANDLE_VALUE;
        PDETOUR_BINARY pBinary = NULL;

        CHAR szOrg[MAX_PATH];
        CHAR szNew[MAX_PATH];
        CHAR szOld[MAX_PATH];

        szOld[0] = '\0';
        szNew[0] = '\0';

        StringCchCopyA(szOrg, sizeof(szOrg), pszPath);
        StringCchCopyA(szNew, sizeof(szNew), szOrg);
        StringCchCatA(szNew, sizeof(szNew), "#");
        StringCchCopyA(szOld, sizeof(szOld), szOrg);
        StringCchCatA(szOld, sizeof(szOld), "~");
        printf("  %s:\n", pszPath);

        hOrg = CreateFileA
        (        szOrg
 , // LPCTSTR lpFileName,
                GENERIC_READ                        , // DWORD
dwDesiredAccess,
                FILE_SHARE_READ                       , // DW
ORD dwShareMode,
                NULL                                        ,
// LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                OPEN_EXISTING                        , // DWOR
D dwCreationDisposition,
                FILE_ATTRIBUTE_NORMAL        , // DWORD dwFlag
sAndAttributes,
```

```
                NULL
 // HANDLE hTemplateFile
        );
        if (hOrg == INVALID_HANDLE_VALUE)
        {
                printf("Couldn't open input file: %s, error: %
d\n",
                        szOrg, GetLastError());
                bGood = FALSE;
                goto end;
        }

        hNew = CreateFileA
        (
                szNew
                                                        , // LP
CTSTR lpFileName,
                GENERIC_WRITE | GENERIC_READ
                             , // DWORD dwDesiredAccess,
                0
                                                          ,
// DWORD dwShareMode,
                NULL
                                                    , // LPSE
CURITY_ATTRIBUTES lpSecurityAttributes,
                CREATE_ALWAYS
                                                , // DWORD dwCr
eationDisposition,
                FILE_ATTRIBUTE_NORMAL | FILE_FLAG_SEQUENTIAL_S
CAN        , // DWORD dwFlagsAndAttributes,
                NULL
                                                        // HAND
LE hTemplateFile
```

```
        );
        if (hNew == INVALID_HANDLE_VALUE) {
                printf("Couldn't open output file: %s, error:
%d\n",
                        szNew, GetLastError());
                bGood = FALSE;
                goto end;
        }

        if ((pBinary = DetourBinaryOpen(hOrg)) == NULL) {
                printf("DetourBinaryOpen failed: %d\n", GetLas
tError());
                goto end;
        }

        if (hOrg != INVALID_HANDLE_VALUE) {
                CloseHandle(hOrg);
                hOrg = INVALID_HANDLE_VALUE;
        }

        {
                BOOL bAddedDll = FALSE;

                DetourBinaryResetImports(pBinary);

                if (!s_fRemove) {
                        if (!DetourBinaryEditImports(pBinary,
                                &bAddedDll,
                                AddBywayCallback, NULL, NULL,
NULL)) {
```

```
                                printf("DetourBinaryEditImport
s failed: %d\n", GetLastError());
                        }
                }

                if (!DetourBinaryEditImports(pBinary, NULL,
                        ListBywayCallback, ListFileCallback,
                        NULL, NULL)) {

                        printf("DetourBinaryEditImports faile
d: %d\n", GetLastError());
                }

                if (!DetourBinaryWrite(pBinary, hNew)) {
                        printf("DetourBinaryWrite failed: %d
\n", GetLastError());
                        bGood = FALSE;
                }

                DetourBinaryClose(pBinary);
                pBinary = NULL;

                if (hNew != INVALID_HANDLE_VALUE) {
                        CloseHandle(hNew);
                        hNew = INVALID_HANDLE_VALUE;
                }

                if (bGood) {
                        if (!DeleteFileA(szOld)) {
                                DWORD dwError = GetLastError
();
```

```
                                if (dwError != ERROR_FILE_NOT_
FOUND) {
                                        printf("Warning: Could
n't delete %s: %d\n", szOld, dwError);
                                        bGood = FALSE;
                                }
                        }
                        if (!MoveFileA(szOrg, szOld)) {
                                printf("Error: Couldn't back u
p %s to %s: %d\n",
                                        szOrg, szOld, GetLastE
rror());
                                bGood = FALSE;
                        }
                        if (!MoveFileA(szNew, szOrg)) {
                                printf("Error: Couldn't instal
l %s as %s: %d\n",
                                        szNew, szOrg, GetLastE
rror());
                                bGood = FALSE;
                        }
                }

                DeleteFileA(szNew);
        }


end:
        if (pBinary) {
                DetourBinaryClose(pBinary);
                pBinary = NULL;
```

```
        }
        if (hNew != INVALID_HANDLE_VALUE) {
                CloseHandle(hNew);
                hNew = INVALID_HANDLE_VALUE;
        }
        if (hOrg != INVALID_HANDLE_VALUE) {
                CloseHandle(hOrg);
                hOrg = INVALID_HANDLE_VALUE;
        }
        return bGood;
}
```

2-1) 원본 EXE import table

```
Dump of file sleep5.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
                412000 Import Address Table
                417D94 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                  57D Sleep
                  611 WriteConsoleW
                  44D QueryPerformanceCounter
                  218 GetCurrentProcessId
                  21C GetCurrentThreadId
                  2E9 GetSystemTimeAsFileTime
```

```
363 InitializeSListHead
37F IsDebuggerPresent
5AD UnhandledExceptionFilter
56D SetUnhandledExceptionFilter
2D0 GetStartupInfoW
386 IsProcessorFeaturePresent
278 GetModuleHandleW
217 GetCurrentProcess
58C TerminateProcess
4D3 RtlUnwind
261 GetLastError
532 SetLastError
131 EnterCriticalSection
3BD LeaveCriticalSection
110 DeleteCriticalSection
35F InitializeCriticalSectionAndSpinCount
59E TlsAlloc
5A0 TlsGetValue
5A1 TlsSetValue
59F TlsFree
1AB FreeLibrary
2AE GetProcAddress
3C3 LoadLibraryExW
462 RaiseException
2D2 GetStdHandle
612 WriteFile
274 GetModuleFileNameW
15E ExitProcess
277 GetModuleHandleExW
1D6 GetCommandLineA
```

```
1D7 GetCommandLineW
345 HeapAlloc
349 HeapFree
 9B CompareStringW
3B1 LCMapStringW
24E GetFileType
175 FindClose
17B FindFirstFileExW
18C FindNextFileW
38B IsValidCodePage
1B2 GetACP
297 GetOEMCP
1C1 GetCPInfo
3EF MultiByteToWideChar
5FE WideCharToMultiByte
237 GetEnvironmentStringsW
1AA FreeEnvironmentStringsW
514 SetEnvironmentVariableW
54A SetStdHandle
2D7 GetStringTypeW
2B4 GetProcessHeap
19F FlushFileBuffers
1EA GetConsoleCP
1FC GetConsoleMode
24C GetFileSizeEx
523 SetFilePointerEx
34E HeapSize
34C HeapReAlloc
 86 CloseHandle
 CB CreateFileW
```

```
                  109 DecodePointer

  Summary

        2000 .data
        7000 .rdata
        1000 .reloc
       11000 .text
```

2-2) 후킹된 EXE import table

```
Dump of file sleep5.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    D:\SecuLetter\git\Detours\bin.X86\simple32.dll
                41C258 Import Address Table
                41C140 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                       Ordinal     1

    KERNEL32.dll
                412000 Import Address Table
                41C148 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                  57D Sleep
```

```
                611 WriteConsoleW
                44D QueryPerformanceCounter
                218 GetCurrentProcessId
                21C GetCurrentThreadId
                2E9 GetSystemTimeAsFileTime
                363 InitializeSListHead
                37F IsDebuggerPresent
                5AD UnhandledExceptionFilter
                56D SetUnhandledExceptionFilter
                2D0 GetStartupInfoW
                386 IsProcessorFeaturePresent
                278 GetModuleHandleW
                217 GetCurrentProcess
                58C TerminateProcess
                4D3 RtlUnwind
                261 GetLastError
                532 SetLastError
                131 EnterCriticalSection
                3BD LeaveCriticalSection
                110 DeleteCriticalSection
                35F InitializeCriticalSectionAndSpinCount
                59E TlsAlloc
                5A0 TlsGetValue
                5A1 TlsSetValue
                59F TlsFree
                1AB FreeLibrary
                2AE GetProcAddress
                3C3 LoadLibraryExW
                462 RaiseException
                2D2 GetStdHandle
```

```
                612 WriteFile
                274 GetModuleFileNameW
                15E ExitProcess
                277 GetModuleHandleExW
                1D6 GetCommandLineA
                1D7 GetCommandLineW
                345 HeapAlloc
                349 HeapFree
                 9B CompareStringW
                3B1 LCMapStringW
                24E GetFileType
                175 FindClose
                17B FindFirstFileExW
                18C FindNextFileW
                38B IsValidCodePage
                1B2 GetACP
                297 GetOEMCP
                1C1 GetCPInfo
                3EF MultiByteToWideChar
                5FE WideCharToMultiByte
                237 GetEnvironmentStringsW
                1AA FreeEnvironmentStringsW
                514 SetEnvironmentVariableW
                54A SetStdHandle
                2D7 GetStringTypeW
                2B4 GetProcessHeap
                19F FlushFileBuffers
                1EA GetConsoleCP
                1FC GetConsoleMode
                24C GetFileSizeEx
```

```
                  523 SetFilePointerEx
                  34E HeapSize
                  34C HeapReAlloc
                   86 CloseHandle
                   CB CreateFileW
                  109 DecodePointer

  Summary

        2000 .data
        1000 .detour
        7000 .rdata
        1000 .reloc
       11000 .text
```

2-3) Detours Function

```
BOOL DetourEnumerateExports(
    _In_     HMODULE hModule,
    _In_opt_ PVOID pContext,
    _In_     PF_DETOUR_ENUMERATE_EXPORT_CALLBACK pfExport
    );
```

- 모듈에서 내보내기를 열거합니다.
  - hModule : 내보내기를 열거 할 모듈의 핸들입니다.
  - pContext : pfExport로 전달 될 프로그램 특정 컨텍스트.
  - pfExport : 콜백 함수는 모듈에서 내 보낸 심볼 당 한 번 호출됩니다.

```
typedef struct _IMAGE_DOS_HEADER {// DOS .EXE 헤더
    WORD e_magic     ; // 매직 넘버
    WORD e_cblp      ; // 파일의 마지막 페이지에서 바이트
    WORD e_cp        ; // 파일
    WORD e_crlc      ; // 재배치
    WORD e_cparhdr   ; // 단락의 헤더 크기
```

```
    WORD e_minalloc ; // 최소 추가 단락 필요
    WORD e_maxalloc ; // 최대 추가 단락 필요
    WORD e_ss       ; // 초기 (상대) SS 값
    WORD e_sp       ; // 초기 SP 값
    WORD e_csum     ; // 체크섬
    WORD e_ip       ; // 초기 IP 값
    WORD e_cs       ; // 초기 (상대) CS 값
    WORD e_lfarlc   ; // 재배치 테이블의 파일 주소
    WORD e_ovno     ; // 오버레이 번호
    WORD e_res [4]  ; // 예약어
    WORD e_oemid    ; // OEM 식별자 (e_oeminfo 용)
    WORD e_oeminfo  ; // OEM 정보; e_oemid 특정
    WORD e_res2 [10]; // 예약어
    LONG e_lfanew   ; // 새로운 exe 헤더의 파일 주소
  } IMAGE_DOS_HEADER, * PIMAGE_DOS_HEADER;
```

- e_magic : DOS 헤더를 구별하는 식별자.
  - 해당 값을 확인하여 올바른 MZ파일인지 검사
    - 모든 실행파일은 파일 가장 첫부분에 'MZ'라는 2바이트의 아스키코드 값을 가지고 있다.
  - 식별 상수 : IMAGE_DOS_SIGNATURE

```
 hFile=CreateFile(filepath,GENERIC_READ,FILE_SHARE_READ,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
 hFileMap=CreateFileMapping(hFile,NULL,PAGE_READONLY,0,0,NULL);
// filepath의 경로의 핸들을 가져오기위한
 dwsize=GetFileSize(hFile,0);
// 가져온 핸들의 파일 크기를 가져온다
 pBaseFile=(char *)MapViewOfFile(hFileMap,FILE_MAP_READ,0,0,dwsize);
// 메모리를 공유하기 위해 메모리에 적재 하고 주소를 pBaseFile에 넘겨준다.
 PIMAGE_DOS_HEADER pIDH=(PIMAGE_DOS_HEADER)pBaseFile;
```

```
// pBaseFile이 char형 이므로 확장시켜 pIDH에 넣어주면 파일의 IMAGE_DOS_HEADER을 얻을수 있다.
```

```
PDETOUR_BINARY DetourBinaryOpen (
    _In_ HANDLE hFile
    );
```

- 바이너리의 내용을 편집하기 위해 메모리로 읽습니다.
  - hFile : 편집을 위해 Create로 파일오픈한 핸들

```
BOOL DetourBinaryResetImports (
    _In_ PDETOUR_BINARY pBinary
    );
```

- 바이너리의 Import table을 제거한다.
  - pBinary : DetourBinaryOpen()함수로 연 바이너리 핸들

```
BOOL DetourBinaryEditImports(
    _In_     PDETOUR_BINARY pBinary,
    _In_opt_ PVOID pContext,
    _In_opt_ PF_DETOUR_BINARY_BYWAY_CALLBACK pfByway,
    _In_opt_ PF_DETOUR_BINARY_FILE_CALLBACK pfFile,
    _In_opt_ PF_DETOUR_BINARY_SYMBOL_CALLBACK pfSymbol,
    _In_opt_ PF_DETOUR_BINARY_COMMIT_CALLBACK pfFinal
    );
```

- Binary의 Import table을 편집하는 함수
  - pBinary : DetourBinaryOpen()함수로 연 바이너리 핸들
  - *pContext* : 각 콜백 함수에 수정없이 전달 될 프로그램 특정 컨텍스트 포인터.
  - *pfByway* : Import table의 각 모듈 전에 콜백 함수가 호출되었습니다.
  - *pfFile* : Import table의 각 모듈에 대해 콜백 함수가 한 번 호출되었습니다.
  - *pfSymbol* : Import table의 각 심볼에 대해 콜백 함수가 한 번 호출되었습니다.
  - *pfCommit* : 오류가없는 경우 Import table의 끝에서 콜백 함수가 호출되었습니다.

```
BOOL BinaryBywayCallback(
    _In_opt_                  PVOID pContext,
    _In_opt_                  LPCSTR pszFile,
    _Outptr_result_maybenull_ LPCSTR * ppszOutFile
```

```
    );
```

- DetourBinaryEditImports API를 사용하여 Import 테이블을 편집하는 동안 기존의 각 Byway에 대해 한 번만 호출되거나 새 Byway를 삽입 할 수 있는 기능을 가리키는 포인터
  - pContext : 수정된 프로그램 특정 컨텍스트 포인터는 해당 인수로 전달됩니다.
  - pszFile : 현재 Import table에 나열된 Byway 이름 또는 NULL.
  - ppszOutFile : 원하는 Byway 출력 이름을 가리키는 포인터
- Byway로 삽입 된 각 DLL은 서수 # 1의 함수를 내 보내야합니다. DLL에 대한 내보내기 테이블에서 서 수가 # 1 인 함수를 내 보내지 않으면 대상 바이너리가 올바르게로드되지 않습니다.

```
BOOL BinaryFileCallback(
    _In_opt_                    PVOID pContext,
    _In_                        LPCSTR pszOrigFile,
    _In_                        LPCSTR pszFile,
    _Outptr_result_maybenull_   LPCSTR * ppszOutFile
    );
```

- DetourBinaryEditImports API를 사용하여 Import 테이블을 편집하는 동안 파일에 대해 한 번만 호출되는 함수를 가리키는 포인터.
  - *pContext* 님의 pContext 인수로 전달 비 개질 된 프로그램 특정 컨텍스트 포인터 DetourBinaryEditImports .
  - *pszOrigFile* : 원본 가져 오기 테이블에 나열된 파일 이름입니다.
  - *pszFile* : 현재 가져 오기 테이블에 나열된 파일 이름
  - *ppszOutFile* : 원하는 가져 오기 테이블 이름을 출력하기위한 포인터.

```
BOOL BinarySymbolCallback(
    _In_opt_                    PVOID pContext,
    _In_                        ULONG nOrigOrdinal,
    _In_                        ULONG nOrdinal,
    _Out_                       ULONG * pnOutOrdinal,
    _In_opt_                    PCSTR pszOrigSymbol,
    _In_opt_                    PCSTR pszSymbol,
    _Outptr_result_maybenull_   PCSTR *ppszOutSymbol
    );
```

- DetourBinaryEditImports API를 사용하여 Import table을 편집하는 동안 각 심볼에 대해 한 번 호출되는 함수를 가리키는는 포인터

- *pContext* 님의 pContext 인수로 전달 비 개질 된 프로그램 특정 컨텍스트 포인터 DetourBinaryEditImports .
- *nOrigOrdinal* : 원본 가져 오기 테이블에 나열된 가져 오기 서수.
- *nOrdinal* : 현재 가져 오기 테이블에 나열된 가져 오기 서수.
- *pnOutOrdinal* : 원하는 가져 오기 서수를 출력하기위한 포인터.
- *pszOrigSymbol* : 원본 가져 오기 테이블에 나열된 가져 오기 기호.
- *pszSymbol* : 현재 가져 오기 테이블에 나열된 가져 오기 기호.
- *ppszOutSymbol* : 원하는 가져 오기 기호를 출력하기위한 포인터.

```
BOOL BinaryCommitCallback(
    _In_opt_ PVOID pContext
    );
```

- DetourBinaryEditImports API를 사용하여 Import table 편집이 끝날 때 호출되는 함수를 가리키는 포인터
  - pContext : 수정 된 프로그램 특정 컨텍스트 포인터는 pContext 인수로 DetourBinaryEditImports에 전달

```
BOOL DetourBinaryWrite(
    _In_ PDETOUR_BINARY pBinary,
    _In_ HANDLE hFile
    );
```

- 업데이트된 바이너를 파일에 씁니다.
  - *pBinary* : 파일에 쓸 바이너리를 가리키는 포인터.
  - *hFile* : 바이너리의 내용을받을 파일의 핸들.

---

3) Hooking DLL

- 이 DLL은 Windows SleepEx API를 우회하여 TimedSleep 함수 대신 호출됩니다. TimedSleepEx는 이전 및 이후 시간을 기록하고 TrueSleepEx 함수 포인터를 통해 실제 SleepEx API를 호출합니다.

3-1) Header

```
#pragma once


#include "framework.h"


#include "../../../../../cppcore/Inc/cppcore.h"
#pragma comment(lib, "cppcore.lib")
```

```
#define _X86_
#include "windef.h"
#include "WinBase.h"
#include "detours.h"
#pragma comment(lib, "detours.lib")

//DWORD __declspec(dllexport) WINAPI TimedSleepEx(DWORD dwMill
iseconds, BOOL bAlertable);
```

```
; HookingDll.def : DLL에 대한 모듈 매개 변수를 정의 합니다.
LIBRARY "HookingDll"
EXPORTS
DetourFinishHelperProcess @1 NONAME
TimedSleepEx @2
```

- DLLName.def File 생성
- DLL 속성 → 링커 → 입력 → 모듈 정의 파일 : DLLName.def 추가
  - LIBRARY [library][BASE=address]
    - 링커로 하여금 DLL 을 만들도록 지시한다.
    - [library] : DLL 의 이름을 지정한다.
    - [BASE=address] : IMAGE_OPTIONAL_HEADER 의 ImageBase 필드 값을 지정한다.
  - entryname[=internalname] [@ordinal [NONAME]] [PRIVATE] [DATA]
    - Export 할 함수에 대해 정의한다.
    - entryname : Export 되는 함수의 이름을 지정한다.
      - [=internalname] : 함수의 원래 이름과 다른 이름으로 Export 하고자 할 때에 사용하며entryname 에는 변경하고 싶은 이름을, internalname 에는 함수의 원래 이름을 지정한다.
        로드하고 있는 DLL 의 함수로 대처하는 것도 가능하며 "DLL파일이름"." 함수이름" 으로 표기한다.
    - [@ordinal] : DLL 의 Export Table 에 함수 이름이 아닌 번호가 들어가도록 지정한다.
    - [NONAME] : 번호를 통해서만 함수를 Export 하도록 지정한다. (함수 이름으로 GetProcAddress 를 호출 시 실패한다)

- [PRIVATE] : .Lib 파일의 Export 함수 정보에서 해당 함수정보를 제거한다.
- [DATA] : 데이터 변수를 익스포트 할 시에 사용한다.

3-2) dllMain.cpp

```
#include "pch.h"


static LONG dwSlept = 0;
static DWORD(WINAPI* TrueSleepEx)(DWORD dwMilliseconds, BOOL b
Alertable) = SleepEx;


DWORD WINAPI TimedSleepEx(DWORD dwMilliseconds, BOOL bAlertabl
e)
{
        DWORD dwBeg = GetTickCount();
        DWORD ret = TrueSleepEx(dwMilliseconds, bAlertable);
        DWORD dwEnd = GetTickCount();

        InterlockedExchangeAdd(&dwSlept, dwEnd - dwBeg);

        return ret;
}


BOOL WINAPI DllMain(HINSTANCE hinst, DWORD dwReason, LPVOID re
served)
{
        LONG error;
        (void)hinst;
        (void)reserved;

        if (DetourIsHelperProcess())
        {
                return TRUE;
```

```
        }

        if (dwReason == DLL_PROCESS_ATTACH)
        {
                DetourRestoreAfterWith();

                printf("simple" DETOURS_STRINGIFY(DETOURS_BIT
S) ".dll:"
                        " Starting.\n");
                fflush(stdout);

                DetourTransactionBegin();
                DetourUpdateThread(GetCurrentThread());
                DetourAttach(&(PVOID&)TrueSleepEx, TimedSleepE
x);
                error = DetourTransactionCommit();

                if (error == NO_ERROR) {
                        printf("simple" DETOURS_STRINGIFY(DETO
URS_BITS) ".dll:"
                                " NBW:Detoured SleepEx().\n");
                }
                else {
                        printf("simple" DETOURS_STRINGIFY(DETO
URS_BITS) ".dll:"
                                " Error detouring SleepEx(): %
d\n", error);
                }
        }
        else if (dwReason == DLL_PROCESS_DETACH) {
                DetourTransactionBegin();
```

```
                DetourUpdateThread(GetCurrentThread());
                DetourDetach(&(PVOID&)TrueSleepEx, TimedSleepE
x);
                error = DetourTransactionCommit();

                printf("simple" DETOURS_STRINGIFY(DETOURS_BIT
S) ".dll:"
                       " Removed SleepEx() (result=%d), slept
%d ticks.\n", error, dwSlept);
                fflush(stdout);
        }
        return TRUE;
}
```

3-3) HookingDll Export Table

```
D:\SecuLetter\git\lockdown.sle\Build\Win32DebugMT>dumpbin /exp
orts HookingDll.dll
Microsoft (R) COFF/PE Dumper Version 14.16.27034.0
Copyright (C) Microsoft Corporation.  All rights reserved.



Dump of file HookingDll.dll

File Type: DLL

  Section contains the following exports for HookingDll.dll

    00000000 characteristics
    FFFFFFFF time date stamp
        0.00 version
           1 ordinal base
           1 number of functions
```

```
           1 number of names

    ordinal hint RVA      name

          1    0 0004E38C ?TimedSleepEx@@YGKKH@Z = @ILT+9095(?
TimedSleepEx@@YGKKH@Z)

  Summary

        1000 .00cfg
        3000 .data
        2000 .detourc
        1000 .detourd
        1000 .idata
        1000 .msvcjmc
       24000 .rdata
        6000 .reloc
        1000 .rsrc
       A3000 .text
       4B000 .textbss
```

3-4) Detours Function

```
BOOL DetourIsHelperProcess(VOID);
```

- 현재 프로세스가 도우미 프로세스인지 또는 대상 프로세스인지 확인
  - Return
    - TRUE : 이 프로세스가 헬퍼 프로세스인 경우
    - FALSE : 이 프로세스가 대상 프로세스인 경우
  - Desc
    - 64bit 상위 프로세스에서 32bit 대상 프로세스를 작성하거나 그 반대 경우일때, DetourCreateProcessWithDllEx API는 임시 도우미 프로세스를 작성해야 합니다.
    - rundll32.exe 매커니즘을 사용하여 사용자 제공 DLL의 복사본을 도우미 프로세스에 로드합니다.

        - 사용자 제공 DLL은 DllMain 함수 내에서 DetourIsHelperProcess 를 호출하여 도우미 프로세스 또는 대상 프로세스에 로드되었는지 확인해야 합니다.
        - 사용자 제공 DLL은 DetourFinishHelperProcess API를 Ordinal 1로 내보내기 해야 합니다.

```
BOOL DetourRestoreAfterWith(VOID);
```

- DetourCreateProcesWithDllEx or DetourCreateProcessWithDlls로 프로세스가 시작된 후 메모리 Import Table Contents를 복원한다.
    - Return
        - True : 필요한 페이로드가 발견되고 복원이 성공한 경우
    - Desc
        - DetourCreateProcessWithDllEx API는 생성한 새로운 프로세스의 타겟 PE 바이너리 프로그램의 Memory Import Table을 수정한다.
        - 올바른 응용프로그램 호환성을 위해서 실행 전 Import Table의 변경 사항을 제거해야 한다.
        - 이러한 변경 사항을 제거하기위해 DetourCreateProcessWithDllEx API는 DetourCopyPayloadToProcess API를 사용하여 관련 반전 데이터를 대상 프로세스의 페이로드에 복사합니다.
        - 대상 프로세스에서 호출되면 DetourRestoreAfterWith API는 필요한 페이로드를 검색하고 가져오기 테이블의 내용을 복원합니다.

```
LONG DetourTransactionBegin(VOID);
```

- Detours 연결 또는 분리를 위한 새 트랙잰션을 시작한다.
    - Desc
        - 트랜잭션을 시작한 후 프로그램은 DetourAttach 또는 DetourAttachEx API를 호출하여 대상 기능에 우회를 연결하거나 DetourDetach API를 호출하여 대상 기능에서 우회를 분리하거나 DetourUpdateThread API를 호출하여 트랜잭션 업데이트에 스레드를 포함시킵니다.
        - 연결, 분리 및 스레드 작업은 프로그램이 DetourTransactionCommit 또는 DetourTransactionCommitEx API를 사용하여 트랜잭션을 커밋 할 때까지 적용되지 않습니다. 또는 프로그램에서 DetourTransactionAbort API를 사용하여 트랜잭션을 중단 할 수 있습니다.

```
LONG DetourUpdateThread(
    _In_ HANDLE hThread
    );
```

- 현재 트랜잭션에서 업데이트 할 스레드를 참여시킵니다.
    - *hThread* : 보류중인 트랜잭션으로 업데이트 할 스레드의 핸들입니다.
    - Desc

        - DetourTransactionBegin API에 의해 열린 현재 트랜잭션이 커밋 될 때 DetourUpdateThread가 업데이트를 위해 지정된 스레드를 참여시킵니다.
        - 우회 트랜잭션이 종료되면 Detours는 DetourUpdateThread API를 통해 변환에 참여한 모든 스레드가 해당 명령 포인터가 대상 함수 또는 트램펄린 함수에서 다시 작성된 코드 내에있는 경우 업데이트되도록합니다.
        - 트랜잭션에 참여하지 않은 스레드는 트랜잭션이 커밋 될 때 업데이트되지 않습니다. 결과적으로 이전 코드와 새 코드의 불법 조합을 실행하려고 시도 할 수 있습니다.

```
LONG DetourAttach(
    _Inout_ PVOID * ppPointer,
    _In_    PVOID pDetour
    );
```

- 우회를 대상 기능에 연결하십시오.
    - ppPointer : 우회가 연결될 대상 포인터의 포인터.
    - pDetour : 우회 기능의 포인터.
    - Desc
        - DetourAttach는 DetourTransactionBegin API에서 열린 현재 트랜잭션의 일부로 대상 함수에 우회를 연결합니다.

```
LONG DetourDetach(
    _Inout_ PVOID * ppPointer,
    _In_    PVOID pDetour
    );
```

- 대상 기능에서 우회를 분리하십시오.
    - ppPointer : 우회가 분리 될 대상 포인터의 포인터.
    - pDetour : 우회 기능의 포인터.
    - Desc
        - DetourDetach는 DetourTransactionBegin API에 의해 열린 현재 트랜잭션의 일부로 대상 함수에서 우회를 분리합니다.

```
LONG DetourTransactionCommit(VOID);
```

- 현재 트랜잭션을 커밋한다.
    - Desc
        - DetourTransactionCommit은 DetourTransactionBegin으로 만든 현재 트랜잭션을 커밋합니다.
        - 트랜잭션을 커밋하면 트랜잭션 내 DetourAttach, DetourAttachEx, DetourDetach 또는 DetourUpdateThread API 호출에 지정된 모든 업데이트가

수행됩니다.

---

3. WithDll.exe

- DetourCreateProcessWithDlls API를 사용하여 타겟 Application을 수정하지 않고 Detour DLL을 프로세스에 로드하는 샘플
- CreateProcess로 명명된 DLL을 호출하여 대상 프로세스에 로드합니다.

1) Validate DLLs

```
for (DWORD n = 0; n < nDlls; n++)
{
     CHAR szDllPath[1024];
     PCHAR pszFilePart = NULL;

     if (!GetFullPathNameA(rpszDllsRaw[n], ARRAYSIZE(szDllPat
h), szDllPath, &pszFilePart))
     {
      printf("withdll.exe: Error: %s is not a valid path nam
e..\n",
      rpszDllsRaw[n]);
      return 9002;
     }

     DWORD c = (DWORD)strlen(szDllPath) + 1;
     PCHAR psz = new CHAR[c];
     StringCchCopyA(psz, c, szDllPath);
     rpszDllsOut[n] = psz;

     HMODULE hDll = LoadLibraryExA(rpszDllsOut[n], NULL, DONT_
RESOLVE_DLL_REFERENCES);
     if (hDll == NULL)
     {
```

```
       printf("withdll.exe: Error: %s failed to load (error %
d).\n",
                                rpszDllsOut[n],
                                GetLastError());
       return 9003;
     }

     ExportContext ec;
     ec.fHasOrdinal1 = FALSE;
     ec.nExports = 0;
     DetourEnumerateExports(hDll, &ec, ExportCallback);
     FreeLibrary(hDll);

     if (!ec.fHasOrdinal1)
     {
       printf("withdll.exe: Error: %s does not export ordinal
#1.\n",
                                rpszDllsOut[n]);
       printf("             See help entry DetourCreateProcess
WithDllEx in Detours.chm.\n");
       return 9004;
     }
}
```

- DetourEnumerateExports 함수를 통해 Export Function Table내에 Ordinal 1로 Export된 함수가 있는지 판단한다.
  - 없을 경우 에러!
  - Ordinal 1 Function : DetourFinishHelperProcess

```
DWORD dwFlags = CREATE_DEFAULT_ERROR_MODE | CREATE_SUSPENDED;


SetLastError(0);
SearchPathA(NULL, szExe, ".exe", ARRAYSIZE(szFullExe), szFullE
xe, &pszFileExe);
```

```
if (!DetourCreateProcessWithDllsA(szFullExe[0] ? szFullExe : NULL, szCommand,
                NULL, NULL, TRUE, dwFlags, NULL, NULL,
                &si, &pi, nDlls, rpszDllsOut, NULL))
{
  DWORD dwError = GetLastError();
  printf("withdll.exe: DetourCreateProcessWithDllEx failed: %d\n", dwError);
  if (dwError == ERROR_INVALID_HANDLE)
  {
#if DETOURS_64BIT
    printf("withdll.exe: Can't detour a 32-bit target process from a 64-bit parent process.\n");
#else
    printf("withdll.exe: Can't detour a 64-bit target process from a 32-bit parent process.\n");
#endif
  }
  ExitProcess(9009);
}

  if (fVerbose)
  {
    DumpProcess(pi.hProcess);
  }

  ResumeThread(pi.hThread);
  WaitForSingleObject(pi.hProcess, INFINITE);
  DWORD dwResult = 0;
  if (!GetExitCodeProcess(pi.hProcess, &dwResult))
  {
```

```
    printf("withdll.exe: GetExitCodeProcess failed: %d\n", Get
LastError());
    return 9010;
  }

  for (DWORD n = 0; n < nDlls; n++)
  {
    if (rpszDllsOut[n] != NULL)
    {
      delete[] rpszDllsOut[n];
      rpszDllsOut[n] = NULL;
    }
  }

  return dwResult;
```

- DetourCreateProcessWithDllsA 함수를 사용하여 프로세스를 실행시키면서 우회 DLL을 추가한다.

1-1) Detours Function

```
BOOL DetourCreateProcessWithDlls(
    _In_opt_          LPCTSTR lpApplicationName,
    _Inout_opt_       LPTSTR lpCommandLine,
    _In_opt_          LPSECURITY_ATTRIBUTES lpProcessAttribute
s,
    _In_opt_          LPSECURITY_ATTRIBUTES lpThreadAttribute
s,
    _In_              BOOL bInheritHandles,
    _In_              DWORD dwCreationFlags,
    _In_opt_          LPVOID lpEnvironment,
    _In_opt_          LPCTSTR lpCurrentDirectory,
    _In_              LPSTARTUPINFOW lpStartupInfo,
```

```
    _Out_               LPPROCESS_INFORMATION lpProcessInformati
on,
    _In_                DWORD nDlls,
    _In_reads_(nDlls) LPCSTR *rlpDlls,
    _In_opt_            PDETOUR_CREATE_PROCESS_ROUTINEW pfCreate
ProcessW
    );
```

- 새 프로세스를 작성하고 DLL을 로드한다.
  - 타겟 프로세스에 따라 적절한 32bit or 64bit DLL을 선택한다.
  - Param
    - *lpApplicationName* : CreateProcess API에 대해 정의 된 애플리케이션 이름입니다.
    - *lpCommandLine* : CreateProcess API에 대해 정의 된 명령 행입니다.
    - *lpProcessAttributes* : CreateProcess API에 대해 정의 된 프로세스 속성.
    - *lpThreadAttributes* : CreateProcess API에 대해 정의 된 스레드 속성입니다.
    - *bInheritHandles* : 상속은 CreateProcess API에 대해 정의 된대로 플래그를 처리합니다.
    - *dwCreationFlags* : CreateProcess API에 대해 정의 된 작성 플래그.
    - *lpEnvironment* : CreateProcess API에 정의 된 프로세스 환경 변수.
    - *lpCurrentDirectory* : CreateProcess API에 정의 된대로 현재 디렉토리를 처리합니다.
    - *lpStartupInfo* : CreateProcess API에 정의 된대로 프로세스 시작 정보.
    - *lpProcessInformation* : CreateProcess API에 대해 정의 된 프로세스 핸들 정보입니다.
    - *nDlls*는 :에서 DLL을 수 백작 *rlpDlls* .
    - *rlpDlls* : 새 프로세스에 삽입 할 DLL의 경로 이름 배열. 32 비트 및 64 비트 응용 프로그램을 모두 지원하려면 DLL에 32 비트 코드가 포함 된 경우 DLL 이름은 "32"로 끝나고 DLL에 64 비트 코드가 포함 된 경우 "64"로 끝나야합니다. 대상 프로세스의 크기가 상위 프로세스와 다른 경우 우회 경로 경로에서 "32"를 "64"로 또는 "64"를 "32"로 자동 대체합니다.
    - *pfCreateProcessW* : CreateProcess API에 대한 프로그램 특정 대체를 가리키는 포인터. 표준 CreateProcess API를 사용하여 새 프로세스를 작성해야하는 경우 NULL.
  - Desc
    - DetourCreateProcessWithDLLs 지정된 DLL이 삽입 된 새 프로세스를 작성합니다.

- 프로세스는 CREATE_SUSPENDED 플래그가 CreateProcess에 일시 중단 된 상태로 작성됩니다.
- 그런 다음 Detours는 새 프로세스에서 응용 프로그램 바이너리의 이미지를 수정하여 지정된 DLL을 첫 번째 가져오기로 포함하고 실행이 재개됩니다.
- 실행이 재개되면 Windows 프로세스 로더는 먼저 애플리케이션 진입 점을 호출하기 전에 대상 DLL과 어플리케이션의 Import Table에 있는 다른 DLL을 로드합니다.
- DetourCreateProcessWithDlls는 새 프로세스에서 대상 PE 이진 프로그램의 메모리 내 Import Table을 수정합니다.
- 업데이트 된 Import Table에는 대상 DLL에서 내보낸 함수 Ordinal #1에 대한 참조가 포함됩니다.
- 대상 프로세스가 32bit이고 상위 프로세스가 64bit이거나 그 반대 일때, DetourCreateProcessWithDlls는 rundll32.exe를 사용하여 DLL을 도우미 프로세스에 로드합니다.
  - 대상 프로세스 Import Table을 올바른 DLL로 업데이트하기 위해 대상 프로세스를 임시로 일치시키는 작업
- 대상 DLL이 로드된 후 DetourRestoreAfterWith API를 호출하여 메모리 내 Import table의 변경 사항을 되돌릴 수 있습니다.
  - 이러한 변경사항을 쉽게 되돌리기 위해 DetourCreateProcessWithDlls는 DetourCopyPlayloadToProcess API를 사용하여 관련 반전 데이터를 대상 프로세스의 페이로드에 복사합니다.
  - 즉 로드된 DLL은 DetourRestoreAfterWith API를 호출하여 가져오기 테이블의 내용을 복원해야 합니다.

---

4. Direct Memory DLL Injection

```
// dwPID : Target Process PID
BOOL InjectDll(DWORD dwPID, LPCTSTR szDllPath)
{
        HANDLE                  hProcess, hThread;
        LPVOID                  pRemoteBuf;
        DWORD                   dwBufSize = (DWORD)(_tcslen(sz
DllPath) + 1) * sizeof(TCHAR);
        LPTHREAD_START_ROUTINE  pThreadProc;
```

```
        // 파라미터로 받은 프로세스의 핸들을 받아옴.
        if (!(hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALS
E, dwPID)))
        {
                DWORD dwErr = GetLastError();
                printf("OpenProcess(%d) failed!!!\nerror:%d
\n", dwPID, dwErr);
                return FALSE;
        }

        // 해당 프로세스의 가상메모리 공간을 할당 받음.
        // 대상 핸들, 할당할 메모리 번지 지정(NULL이면 시스템이 자동 지
정), 할당할 메모리 양,
        // 할당 방법 지정, 할당한 페이지의 액세스 타입 지정
        // 할당한 메모리 번지 반환 / NULL 반환
        pRemoteBuf = VirtualAllocEx(hProcess, NULL, dwBufSize,
MEM_COMMIT, PAGE_READWRITE);
        if (nullptr == pRemoteBuf)
        {
                DWORD dwErr = GetLastError();
                printf("VirtualAllocEx failure, error:%d\n", d
wErr);
                return FALSE;
        }

        // 해당 프로세스 메모리를 조작.
        // 조작할 대상 프로세스 핸들, 조작할 가상메모리 주소, 메모리에
적을 값(인젝션 시킬 DLL 경로),
        // 메모리에 쓸 크기, 특정 프로세스의 바뀔 바이트를 받는 변수(NU
LL 사용 안함)
        BOOL bRet = WriteProcessMemory(hProcess, pRemoteBuf,
(LPVOID)szDllPath, dwBufSize, NULL);
```

```
        if (FALSE == bRet)
        {
                DWORD dwErr = GetLastError();
                printf("WriteProcessMemory failure, error:%d
\n", dwErr);
                return FALSE;
        }

        pThreadProc = (LPTHREAD_START_ROUTINE)GetProcAddress(G
etModuleHandle(L"kernel32.dll"), "LoadLibraryW");
        if (nullptr == pThreadProc)
        {
                DWORD dwErr = GetLastError();
                printf("GetProcAddress failure, error:%d\n", d
wErr);
                return FALSE;
        }

        hThread = CreateRemoteThread(hProcess, NULL, 0, pThrea
dProc, pRemoteBuf, 0, NULL);
        if (nullptr == hThread)
        {
                DWORD dwErr = GetLastError();
                printf("CreateRemoteThread failure, error:%d
\n", dwErr);
                return FALSE;
        }

        WaitForSingleObject(hThread, INFINITE);

        VirtualFreeEx(hProcess, pRemoteBuf, 0, MEM_RELEASE);
```

```
        CloseHandle(hThread);
        CloseHandle(hProcess);

        return TRUE;
}
```

1) Function

```
HANDLE OpenProcess(
  DWORD dwDesiredAccess,
  BOOL  bInheritHandle,
  DWORD dwProcessId
);
```

- Process의 Handle 값을 얻어온다.
  - Param
    - dwDesiredAccess : 프로세스에 대한 접근 권한 유형
      - PROCESS_ALL_ACCESS : 모든 권한 요청
    - bInheritHandle : PIE로 접근한 프로세스를 현재 이 함수를 실행하고 있는 프로세스에 상속할지 결정하는 인자.
    - dwProcessId : 인자값을 0 or NULL인 경우 모든 Process에 대해 접근

```
LPVOID VirtualAllocEx(
  HANDLE hProcess,
  LPVOID lpAddress,
  SIZE_T dwSize,
  DWORD  flAllocationType,
  DWORD  flProtect
);
```

- 지정된 프로세스의 가상 주소 공간 내에서 메모리 영역의 상태를 예약, 커밋 또는 변경합니다.
- 이 함수는 할당한 메모리를 0으로 초기화합니다.
  - Param
    - hProcess : 해당 프로세스 가상 주소 공간 내 메모리를 할당한다.
      - PROCESS_VM_OPERATION 접근 권한이 있어야 한다.
    - lpAddress : 확보받고 싶은 주소 공간의 시작 주소

        - NULL일 경우 시스템이 자동으로 비어있는 주소를 할당
    - dwSize : 확보받고 싶은 메모리의 크기, 바이트 단위
    - flAllocationType : 메모리 예약(MEM_RESERVE), 실제 물리적 메모리 커밋(MEM_COMMIT)등 메모리 상태 설정
    - flProtect : 메모리에 대한 읽고 쓰기의 접근 범위 설정

```
BOOL WriteProcessMemory(
  HANDLE  hProcess,
  LPVOID  lpBaseAddress,
  LPCVOID lpBuffer,
  SIZE_T  nSize,
  SIZE_T  *lpNumberOfBytesWritten
);
```

- 해당 프로세스 메모리 조작한다.
    - Param
        - hProcess : 대상 프로세스 핸들
        - lpBaseAddress : 해당 프로세스 가상 메모리를 조작할 주소
        - lpBuffer : DLL 경로값
        - nSize : 메모리에 쓸 lpBuffer size
        - lpNumberOfBytesWritten : 매개변수 설정시 이용

```
FARPROC GetProcAddress(
  HMODULE hModule,
  LPCSTR  lpProcName
);
```

- 지정된 DLL에서 Export된 함수또는 변수의 주소를 검색

```
HMODULE GetModuleHandleA(
  LPCSTR lpModuleName
);
```

- 명시된 이름에 해당하는 모듈이 해당 프로세스 주소공간에 매핑되어 있었다면 해당 모듈의 핸들을 반환한다.

```
HANDLE CreateRemoteThread(
  HANDLE                  hProcess,
  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  SIZE_T                  dwStackSize,
```

```
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID                 lpParameter,
  DWORD                  dwCreationFlags,
  LPDWORD                lpThreadId
);
```

- 다른 프로세스에 스레드를 쉽게 생성할 수 있는 함수
  - Param
    - hProcess : 스레드가 작성되는 프로세스에 대한 핸들입니다. 핸들에는 **PROCESS_CREATE_THREAD** , **PROCESS_QUERY_INFORMATION** , **PROCESS_VM_OPERATION** , **PROCESS_VM_WRITE** 및 **PROCESS_VM_READ** 액세스 권한 이 있어야 하며 특정 플랫폼에서 이러한 권한 이 없으면 실패 할 수 있습니다.
    - lpThreadAttributes : 새 스레드의 보안 설명자를 지정하고 자식 프로세스가 반환 된 핸들을 상속 할 수 있는지 여부를 결정 하는 SECURITY_ATTRIBUTES구조에 대한 포인터 입니다.
    - dwStackSize : 스택의 초기 크기 (바이트)입니다.
    - lpStartAddress : 스레드가 실행할 **LPTHREAD_START_ROUTINE** 유형의 응용 프로그램 정의 함수에 대한 포인터 이며 원격 프로세스에서 스레드의 시작 주소를 나타냅니다.
    - lpParameter : 스레드 함수에 전달할 변수에 대한 포인터
    - dwCreationFlags : 스레드 생성을 제어하는 플래그
    - lpThreadId : 스레드 식별자를받는 변수에 대한 포인터입니다.

2) Description

- 대상 프로세스 핸들 구하기
  - OpenProcess API
  - 해당 API를 이용해서 대상 프로세스 핸들을 구한다.
- 대상 프로세스 메모리에 Injection 시킬 DLL 경로 써주기
  - VirtualAllocEx API
    - 대상 프로세스에게 로딩할 DLL 파일의 경로(문자열)를 알려야 한다.
    - 아무 메모리 공간에 쓸 수 없으므로 위 API를 이용하여 메모리 공간에 버퍼를 할당
      - 버퍼 크기는 DLL 경로 문자열 길이(+ NULL, 1byte)
    - VirtualAllocEx API의 반환된 값은 할당된 버퍼의 주소입니다.
      - 해당 주소는 대상 프로세스 핸들이 가리키는 대상의 메모리 주소이다.
  - WriteProcessMemory API
    - 할당 받은 버퍼 주소에 해당 API를 사용하여 DLL 경로 문자열을 써준다.

    - DebugAPI
        - VirtualAllocEx & VirtualFreeEx
        - WriteProcessmemory & ReadProcessMemory
- LoadLibraryA or W API 주소를 구하기
    - GetModuleHandle
        - 현 프로세스(대상 프로세스 X)의 kernel32.dll의 핸들을 구한다.
    - GetProcAddress API
        - 얻어온 핸들을 사용하여 LoadLibrary API의 시작 주소를 얻어온다.
    - 현재 대상 프로세스의 주소가 아니라 본인 프로세스의 주소를 얻어오는데, Windows O/S의 핵심 DLL들은 Relocation이 발생하지 않도록 되어 있다.
        - DLL Injection 기법은 위와 같은 OS 핵심 DLL들은 자신만의 고유한 주소에 로딩되는 것을 보장해주는 Windows 특성을 이용한다.
        - 모든 Windows 프로세스는 kernel32.dll을 로딩합니다.
            - PE Header를 조작하여 IAT에서 해당 DLL을 제거해도 Loader에 의해서 강제적으로 로딩된다.
- 대상 프로세스에 스레드를 실행
    - CreateRemoteThread API
        - LoadLibrary API를 호출하도록 명령만 내리면 되지만, Windows에서는 그런 API가 존재 하지 않는다.
        - 그래서 해당 API를 사용하여 다른 프로세스에게 스레드를 실행시켜준다.

```
HANDLE WINAPI CreateRemoteThread(
  __in   HANDLE                 hProcess,            // 프로세스 핸들
  __in   LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  __in   SIZE_T                 dwStackSize,
  __in   LPTHREAD_START_ROUTINE lpStartAddress,      // 스레드 함수 주소
  __in   LPVOID                 lpParameter,         // 스레드 파라미터 주소
  __in   DWORD                  dwCreationFlags,
  __out  LPDWORD                lpThreadId
);
```

        - ThreadProc 과 LoadLibrary API
            - 다른 프로세스에 DLL을 Injection 시키는데 스레드가 무슨 상관...?
            - 바로! 두함수 모드 4byte 파라미터를 받고 4byte 값을 반환한다!

```
DWORD WINAPI ThreadProc(
  __in  LPVOID           lpParameter
);

HMODULE WINAPI LoadLibrary(
  __in  LPCTSTR          lpFileName
);
```

- CreateRemoteThread API를 호출해서 lpStartAddress에 'LoadLibrary' 주소를 주고 lpParameter에 원하는 'dll 경로'를 주어 실행시킨다.
  - 반드시 대상 프로세스의 가상 메모리 공간에의 주소여야 한다.
- 즉 해당 API는 스레드를 생성하는 것이 아니라 실제로는 LoadLibrary를 호출시키는 것이다.

---

5. [DllMain]

```
BOOL ProcessAttach(HMODULE hDll)
{
        s_bLog = FALSE;
        s_nTlsIndent = TlsAlloc();
        s_nTlsThread = TlsAlloc();
        ThreadAttach(hDll);

        WCHAR wzExeName[MAX_PATH];

        s_hInst = hDll;
        GetModuleFileNameW(hDll, s_wzDllPath, ARRAYSIZE(s_wzDllPath));
        GetModuleFileNameW(NULL, wzExeName, ARRAYSIZE(wzExeName));
        StringCchPrintfA(s_szDllPath, ARRAYSIZE(s_szDllPath), "%ls", s_wzDllPath);
```

```
        ProcessEnumerate();

        LONG error = AttachDetours();
        if (error != NO_ERROR) {
                return FALSE;
        }

        s_bLog = TRUE;
        return TRUE;
}
```

- TlsAlloc() ; TLS : Thread Local Storage
  - 스레드 별로 고유한 저장 공간을 가질 수 있는 방법
  - 스레드 특성
    - 각각의 스레드는 고유한 스택을 갖기 때문에 스택 변수(지역 변수)는 스레드 별로 고유하다.
      - 각각의 스레드가 같은 함수를 실행한다고 해도 그 함수에서 정의된 지역 변수는 실제로 서루 다른 메모리 공간에 위치한다는 의미미
      - 그러나, 정적 변수와 전역 변수의 경우에는 프로세스 내의 모든 스레드에 의해서 공유된다.
  - 개념
    - TLS는 정적, 전역 변수를 각각의 스레드에게 독립적으로 만들어 주고 싶을 때 사용
      - 간은 문장(context)을 실행하고 있지만, 실제로는 스레드 별로 다른 주소 공간을 상대로 작업하는 것
  - 예제
    - 정적 TLS

```
__declspec(thread) int g_nWindows;
```

      - g_nWindows 변수는 모든 스레드에게 고유한(private) 변수이다.
    - 동적 TLS

```
// 공간 확보 (TlsAlloc은 인덱스를 반환하기 전, 해당 블록을 0으로 초기화 시킨다)
DWORD dwIamIndex = ::TlsAlloc();
// 데이터 저장
::TlsSetValue(dwIamIndex, pMyData);
```

```
// 데이터 얻기
BYTE* pGiveMe = (BYTE*)::TlsGetValue(dwIamIndex);
// 공간 해제
::TlsFree(dwIamIndex);
```

- InterlockedIncrement
  - 원형

```
LONG __cdecl InterlockedIncrement(
_Inout_ LONG volatile *Addend);
```

  - 개념
    - 상호자금 함수
    - 여러 스레드에서 공유되는 데이터의 값을 하나 씩 증가시킬때 사용
    - 반대는 InterlockedDecrement()
    - 주의점
      - 만약 값을 2만큼 증가시키기 위해서는 InterlockedIncrement() 함수 2번 호출이 아닌 InterlockedExchangeAdd() 사용
- GetModuleFileName() ; 실행 파일의 경로 얻기
  - 원형

```
DWORD WINAPI GetModuleFileName(
HMODULE hModule,  // 현재 실행되고 있는 모듈의 핸들, 또는 NULL(자신의 실행경로 반환)
LPTSTR IpDileName,  // 실행 경로를 받을 포인터
DWORD nSize  // 실행 경로가 들어갈 버퍼의 길이
);
```

  - 개념
    - 현재 실행되는 프로그램 PullPath가 넘어옴
    - Ex)
      - c://Desktop//TestPJT//Debug//TestPJT.exe
- StringCchPrintf() ; 형식화된 데이터를 지정된 문자열에 쓴다.
  - 원형

```
STRSAFEAPI StringCchPrintfA(
  STRSAFE_LPSTR  pszDest,
  size_t         cchDest,
  STRSAFE_LPCSTR pszFormat,
```

```
  ...
);
```

- PARAM
  - pszDest : 형식화되고, NULL로 종료되는 문자열로 작성된 데이터 저장
  - chchDest : pszDest 버퍼 사이즈