Microsoft Research Detours Package 개요

https://github.com/microsoft/Detours.git

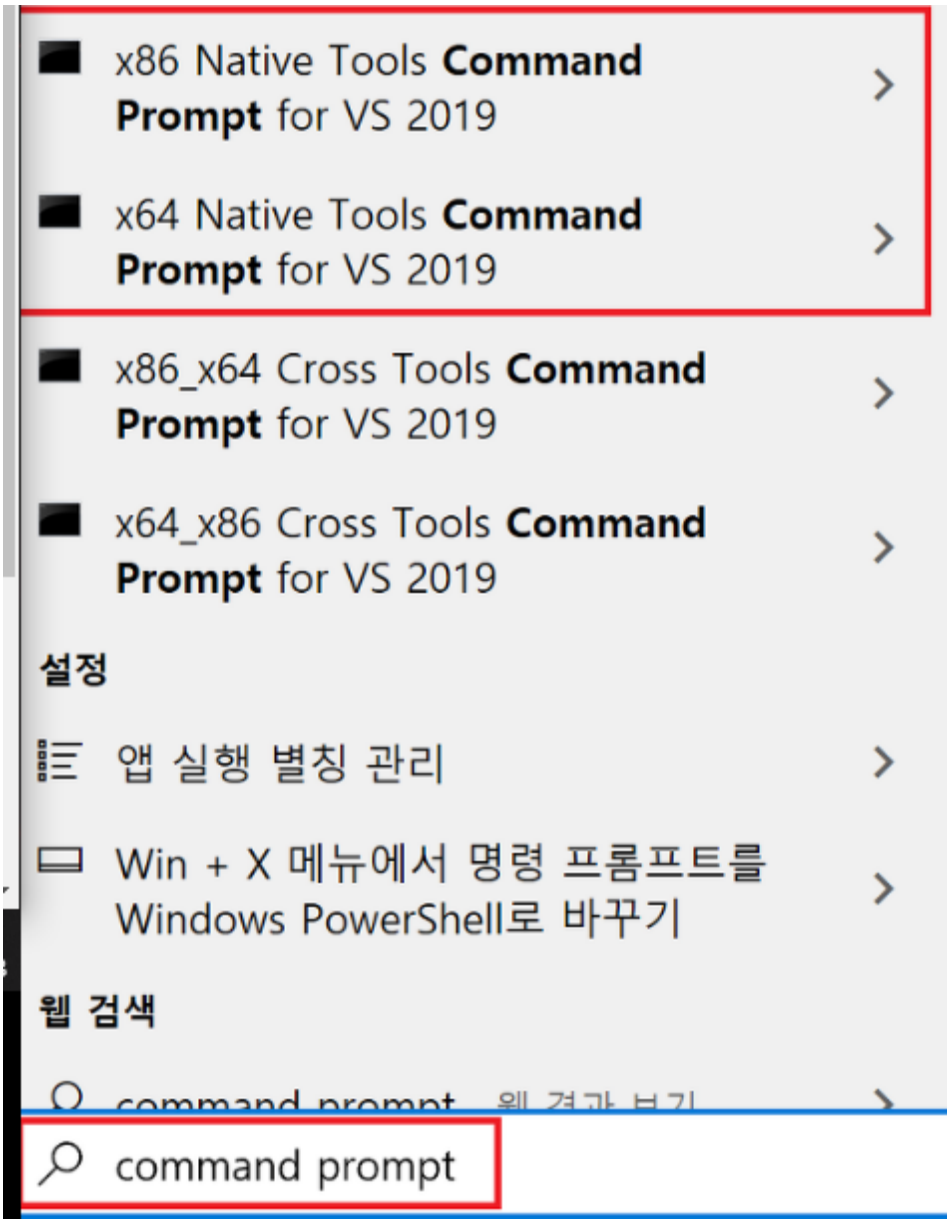

microsoft/Detours • github.com

https://documentation.help/Detours/Detours_Overview.htm

준비

1. Microsoft Detours Build

- Visual Studio 설치
- Command Prompt 실행

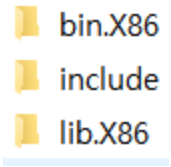


- Detours Source 경로로 이동해서 : nmake 명령 실행

```
nmake
```

```
D:₩SecuLetter₩git₩Detours>nmake
```

2. Build 완료



3. Test

1) setdll

```
D:\SecuLetter\git\lockdown.sle\3rdParty\Detours\bin.X86>setdll
/?
Usage:
    setdll [options] binary_files
Options:
    /d:file.dll   : Add file.dll binary files
    /r            : Remove extra DLLs from binary files
    /?            : This help screen.
```

2) sleep

(1) sleep5

```
D:\SecuLetter\git\lockdown.sle\3rdParty\Detours\bin.X86>sleep5
sleep5.exe: Starting.
sleep5.exe: Done sleeping.
```

(2) setdll 실행

```
D:\SecuLetter\git\lockdown.sle\3rdParty\Detours\bin.X86>setdll
/d:simple32.dll sleep5.exe
Adding simple32.dll to binary files.
  sleep5.exe:
    simple32.dll
    KERNEL32.dll -> KERNEL32.dll
```

(3) sleep5

```
D:\SecuLetter\git\lockdown.sle\3rdParty\Detours\bin.X86>sleep5
simple32.dll: Starting.
simple32.dll: Detoured SleepEx().
sleep5.exe: Starting.
```

```
sleep5.exe: Done sleeping.
simple32.dll: Removed SleepEx() (result=0), slept 5000 ticks.
```

(4) setdll 원복

```
D:\SecuLetter\git\lockdown.sle\3rdParty\Detours\bin.X86>setdll
/r:simple32.dll sleep5.exe
Removing extra DLLs from binary files.
  sleep5.exe:
    KERNEL32.dll -> KERNEL32.dll
```

(5) sleep5

```
D:\SecuLetter\git\lockdown.sle\3rdParty\Detours\bin.X86>sleep5
sleep5.exe: Starting.
sleep5.exe: Done sleeping.
```

Wiki

1. Interception of Binary Functions

1) Description

- Interception code is applied dynamically at runtime.
  - 차단 코드는 런타임에 동적으로 적용됩니다.
- Detours(LIB) replaces the first few instructions of the target function with an unconditional jump to the user-provided detour function.
  - Detours (LIB)는 대상 기능의 처음 몇 가지 명령어를 사용자 제공 우회 기능으로의 무조건 점프로 바꿉니다.
  - user-provided detour function : trampoline function
  - Instructions from the target function are preserved in a trampoline function.
    - 대상 기능의 명령은 트램폴린 기능으로 유지됩니다.
- The trampoline consists of the instructions removed from the target function and an unconditional branch to the remainder of the target function.
  - 트램폴린은 대상 기능에서 제거 된 명령과 나머지 대상 기능에 대한 무조건 분기로 구성됩니다.
- When execution reaches the target function, control jumps directly to the user-supplied detour function.

    - 실행이 목표 기능에 도달하면, 제어는 사용자 제공 우회 기능으로 직접 점프합니다
    - The detour function performs whatever interception preprocessing is appropriate.
        - 우회 기능은 적절한 차단 전처리를 수행합니다.
- The detour function can return control to the source function or it can call the trampoline function, which invokes the target function without interception.
    - 우회 기능은 제어를 소스 기능으로 되돌릴 수도 있고 트램폴린 기능을 호출 할 수도 있습니다.
    - When the target function completes, it returns control to the detour function.
        - 대상 기능이 완료되면 제어를 우회 기능으로 되돌립니다.
    - The detour function performs appropriate postprocessing and returns control to the source function.
        - 우회 기능은 적절한 후 처리를 수행하고 제어를 소스 기능으로 되돌립니다.
- Figure 1 shows the logical flow of control for function invocation with and without interception.
  그림 1은 인터 셉션 유무에 따른 함수 호출을위한 논리적 제어 흐름을 보여줍니다.

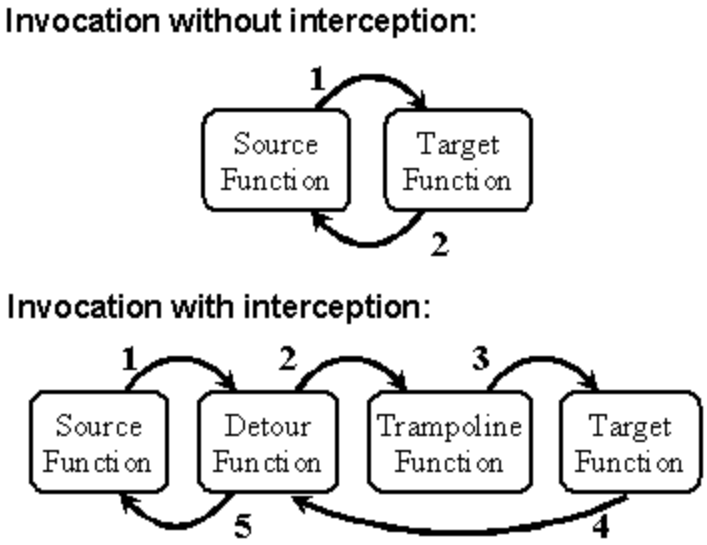


Figure 1. Control flow of invocation without Detours and with Detours.

- The detours library intercepts target functions by rewriting their in-process binary image.
    - Detours 라이브러리는 처리 중 이진 이미지를 다시 작성하여 대상 기능을 가로 챌 수 있습니다.

- For each target function, Detours actually rewrites two functions, the target function and the matching trampoline function, and one function pointer, the target pointer.
  - 각 타겟 기능에 대해, Detours는 실제로 타겟 기능과 일치하는 트램펄린 기능과 하나의 기능 포인터 인 대상 포인터의 두 기능을 다시 작성합니다.
- The trampoline function is allocated dynamically by detours.
  - 트램폴린 기능은 Detours에 의해 동적으로 할당됩니다.
  - Prior to insertion of a detour, the trampoline contains only a single jump instruction to the target function.
    - 우회를 삽입하기 전에 트램폴린에는 대상 기능에 대한 단일 점프 명령 만 포함됩니다.
  - After insertion, the trampoline contains the initial instructions from the target function and a jump to the remainder of the target function.
    - 삽입 후 틀램폴린에는 대상 기능의 초기 명령과 나머지 대상 기능으로의 점프가 포함된다.
- The target pointer is initialized by the user to point to the target function.
  - 대상 포인터는 사용자가 대상 기능을 가리 키도록 초기화됩니다.
  - After a detour is attached to the target function, the target pointer is modified to pint to the trampoline function.
    - Detour가 대상 기능에 부착 된 후 대상 포인터가 트램폴린 기능을 가리 키도록 수정됩니다
  - After the detour is detached from the target function, the target pointer is returned to point to the original target function.
    - Detour가 대상 기능에서 분리 된 후 원래 대상 기능을 가리 키도록 대상 포인터가 반환됩니다.

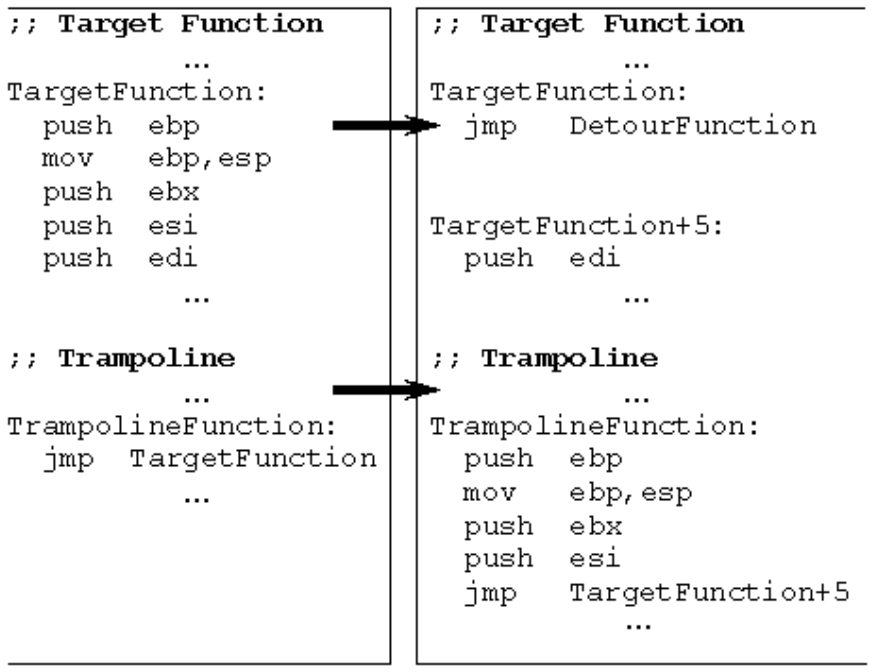


Figure 2. Trampoline and target functions, before (on the left) and after (on the right) insertion of the detour.

- Figure 2 shows the insertion of a detour.
  - Detour의 삽입을 보여준다.
  - To detour a target function, Detours first allocates memory for the dynamic trampoline function (if no static trampoline is provided) and then enables write access to both the target and the trampoline.
    - 대상 기능을 우회하기 위해 Detours는 먼저 동적 트램폴린 기능 (정적 트램펄린이 제공되지 않은 경우)에 메모리를 할당 한 다음 대상과 트램펄린 모두에 대한 쓰기 액세스를 가능하게합니다.
  - Starting with the first instruction, Detours copies instructions from the target to the trampoline until at least 5 Bytes have been copied (enough for an unconditional jump instruction).
    - 첫 번째 명령부터 시작하여 Detours는 최소 5 바이트가 복사 될 때까지 (무조건적인 점프 명령의 경우에도) 대상에서 트램펄린으로 명령을 복사합니다.
    - If the target function is fewer than 5 bytes, Detours aborts and returns an error code.
      - 대상 기능이 5 바이트 미만이면 우회가 중단되고 오류 코드가 반환됩니다.
  - To copy instructions, Detours uses a simple table-driven disassembler.
    - 명령을 복사하기 위해 Detours는 간단한 테이블 구동 디스어셈블러를 사용합니다.

    - Detours adds a jump instruction from the end of the trampoline to the first non-copied instruction of the function.
        - Detours는 트램폴린의 끝에서 대상 함수의 복사되지 않은 첫 번째 명령으로 점프 명령을 추가합니다.
    - Detours writes an unconditional jump instruction to the detour function as the first instruction of the target function.
        - Detours는 대상 기능의 첫 번째 명령으로 무조건 점프 명령을 우회 기능에 씁니다.
    - To finish, Detours restores the original page permissions on both the target and trampoline functions and flushes the CPU instruction cache with a call to the FlushInstructionCache API.
        - 완료하기 위해 Detours는 대상 및 트램펄린 기능 모두에 대한 원래 페이지 권한을 복원하고 FlushInstructionCache API를 호출하여 CPU 명령 캐시를 초기화합니다.

2. Using Detours

- Two things are necessary in order to detour a target function.
    - 대상 기능을 우회하려면 두 가지가 필요합니다.
    - a target pointer containing the address of the target function and a detour function.
        - Target Function의 주소를 포함한 Target Pointer와 Detour Function
- For proper interception the target function, detour function, and the target pointer must have exactly the same call signature including number of arguments and calling convention.
    - 적절한 가로 채기를 위해서는 대상 함수, 우회 함수 및 대상 포인터가 인수 수 및 호출 규칙을 포함하여 정확히 동일한 호출 서명을 가져야합니다.
    - Using the same calling convention insures that registers will be properly preserved and that the stack will be properly aligned between detour and target functions.
        - 동일한 호출 규칙을 사용하면 레지스터가 올바르게 유지되고 스택이 우회 기능과 대상 기능 사이에 올바르게 정렬됩니다.
- The code fragment in Figure 5 illustrates usage of the detours library.
    - 그림 5의 코드 조각은 Detours 라이브러리의 사용법을 보여줍니다.
- User code must include the `detours.h` header file and link with the `detours.lib` library.

```
#include <windows.h>
#include <detours.h>
```

```
static LONG dwSlept = 0;

// Target pointer for the uninstrumented Sleep API.
//
static VOID (WINAPI * TrueSleep)(DWORD dwMilliseconds) = Slee
p;

// Detour function that replaces the Sleep API.
//
VOID WINAPI TimedSleep(DWORD dwMilliseconds)
{
    // Save the before and after times around calling the Slee
p API.
    DWORD dwBeg = GetTickCount();
    TrueSleep(dwMilliseconds);
    DWORD dwEnd = GetTickCount();

    InterlockedExchangeAdd(&dwSlept, dwEnd - dwBeg);
}

// DllMain function attaches and detaches the TimedSleep detou
r to the Sleep target function.
// The Sleep target function is referred to through the TrueSl
eep target pointer.
BOOL WINAPI DllMain(HINSTANCE hinst, DWORD dwReason, LPVOID re
served)
{
    if (DetourIsHelperProcess()) {
        return TRUE;
    }
```

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

        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourAttach(&(PVOID&)TrueSleep, TimedSleep);
        DetourTransactionCommit();
    }
    else if (dwReason == DLL_PROCESS_DETACH) {
        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourDetach(&(PVOID&)TrueSleep, TimedSleep);
        DetourTransactionCommit();
    }
    return TRUE;
}
# Figure 5. simple detour to modify the Windows Sleep API.
```

- Interception of the target function is enabled by invoking the DetourAttach API within a detour transaction.
  - 우회 트랜잭션 내에서 DetourAttach API를 호출하여 대상 기능을 가로 챌 수 있습니다.
  - A detour transaction is marked by calls to the DetourTransactionBegin API and the DetourTransactionCommit API.
    - Detour 트랙젝션은 DetourTransactionBegine 및 Detour TransactionCommit API 호출로 표시됩니다.
    - The DetourAttach API takes two arguments
      - The address of the target pointer and pointer to the detourfunction.
        - 대상 포인터의 주소 및 우회 함수에 대한 포인터
      - The target function is not given as an argument because it must already be stored in the target pointer.
        - 타겟 함수는 타겟 포인터에 이미 저장되어 있어야하므로 인수로 제공되지 않습니다.

- The DetourUpdateThread API enlists threads in the transaction so that their instruction pointers are appropriately updated when the transaction commits.
  - DetourUpdateThread API는 트랙잭션이 커밋 될 때 해당 명령 포인터가 적절히 업데이트되도록 트랜잭션에 스레드를 참여시킵니다.
- The DetourAttach API Allocates and prepares a trampoline for calling the target function.
  - DEtourAttach API는 타겟 함수를 호출하기 위한 트램펄린을 할당하고 준비합니다.
  - When the detour transaction commits, the target function and trampoline are rewritten and the target pointer is updated to point to the trampoline function.
    - Detour 트랜잭션이 커밋되면, 타겟 함수와 트램펄린이 다시 작성되고 대상 포인터가 트램펄린 기능을 가리키도록 업데이트됩니다.
- Once a target function has been detoured, any call to the target function will be re-routed through the detour function.
  - 타겟 함수가 우회되면, 타겟 함수에 대한 모든 호출을 우회 함수를 통해 재 경유하게 됩니다.
  - It is the responsibility of the detour function to copy arguments when invoking the target function through the trampoline.
    - 트램펄린을 통해 타겟 함수를 호출 할 때 인수를 복사하는 것은 우회 함수의 책임입니다.
  - This is intuitive as the target function becomes simply a subroutine callable by the detour function.
    - 타겟 함수가 단순히 우회 함수에 의해 호출 가능한 서부 루틴이되므로 직관적입니다.
- Interception of a target function is removed by calling the DetourDetach API within a detour transaction.
  - Detour 트랜잭션 내에서 DetourDetach API를 호출하면 대상 함수의 차단 기능이 제거됩니다.
  - Like the DetourAttach API, the DetourDetach API takes two arguments
    - DetourAttach API와 마찬가지로 DetourDetach API는 두 가지 인수를 사용합니다.
    - The Address of Target pointer and the pointer to the detour function.
      - 타겟의 포인터의 주소와 우회 함수의 포인터
  - When the detour transaction commits, the target function is rewritten and restored to its original code, the trampoline function is deleted, and the target pointer is restored to point to the original target function.
    - 우회 트랜잭션이 커밋되면 타겟 함수가 다시 작성되어 원래 코드로 복원되고, 트램펄린 함수가 삭제되며, 타겟 포인터가 원래 타겟 함수를 가리키도록 복원됩니다.

- In cases where detour functions need to inserted into an existing application without source code access, the detour functions should be packaged in a DLL.
  - 소스코드 액세스 없이 기존 어플리케이션에 우회 함수를 삽입해야하는 경우 우회 기능을 DLL로 묶어야 합니다.
  - The DLL can be loaded a new process at creation time using the DetourCreateProcessWithDllEx or DetourCreateProcessWithDlls APIs.
    - DLL은 생성시 DetourCreateProcessWithDllEx 또는 DetourCreateProcessWithDlls API를 사용하여 새 프로세스에 로드될 수 있습니다.
  - If a DLL is inserted using DetourCreateProcessWithDllEx or DetourCreateProcessWithDlls, the DllMain function must call the DetourRestoreAfterWith API.
    - 만약 DLL이 DetourCreateProcessWithDllEx 또는 DetourCreateProcessWithDlls를 사용하여 삽입된 경우 DllMain 함수는 DetourRestoreAfterWith API를 호출해야 합니다.
  - If the DLL may be used in mixed 32-bit and 64-bit environments, then the DllMain function must call the DetourIsHelperProcess API.
    - 만약 DLL을 32-bit 및 64-bit 혼합 환경에서 사용할 수 있는 경우 DllMain 함수는 DetourIsHelperProcess API를 호출해야 합니다.
  - The DLL must export the DetourFinishHelperProcess API as export Ordinal 1, which will be called by rundll32.exe to perform the helper tasks.
    - DLL은 DetourFinishHelperProcess API를 내보내기를 서수 1(TRUE)로 내보내야하며, 도우미 작업을 수행하려면 rundll32.exe에 의해 호출됩니다.
- The withdll.exe program include in the Detours package uses the DetourCreateProcessWithDlls API to start a new process with a named DLL.
  - Detours 패키제에 포함된 withdll.exe 프로그램은 DetourCreateProcessWithDlls API를 사용하여 명명된 DLL로 새 프로세스를 시작합니다.

3. Payloads and DLL Import Editing

> *Payload : 사용에 있어서 전송되는 데이터를 뜻한다.*
> *전송의 근본적인 목적이 되는 데이터의 일부분으로 그 데이터와 함께 전송되는 헤더와 메타데이터와 같은 데이터는 제외한다.*

- In addition to APIs for attaching and detaching detours functions, the Detours package also include APIs for attaching arbitrary data segments, called payloads, to windows binary files and for editing DLL import tables.

    - 우회 기능 추가 및 분리를 위한 API 이외에도, Detours 패키지에는 페이로드라는 임의의 데이터 세그멘트를 Windows 이진 파일에 추가하고 DLL Import tables을 편집하기 위한 API가 포함되어 있습니다.
- The binary editing APIs in Detours are fully reversible
    - Detours의 바이너리 편집 API는 완전히 가역적입니다.
    - Detours stores recovery information within the binary to enable removal of the edits at any time in the future.
        - Detours는 바이너리 내에 복구 정보를 저장하여 나중에 언제든지 편집 내용을 제거 할 수 있습니다.

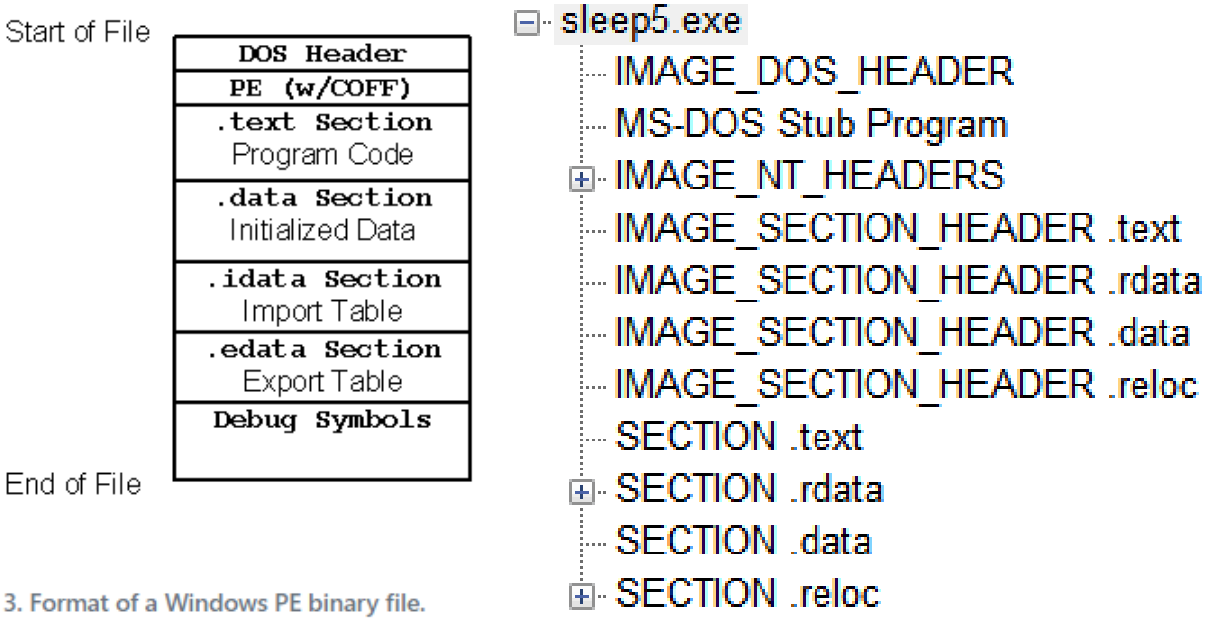


Figure 3. Format of a Windows PE binary file.

- Figure 3 show the basic structure of a Windows Portable Executable(PE) binary file.
- The PE format for Windows binaries is an extension of COFF.
    - Windows 바이너리 PE 형식은 COFF 확장 형식이다.
        - COFF; The Common Object File Format
    - A Windows binary consists of a DOS compatible header, a PE header, a text section containing program code, a data section containing initialized data, an import table listing any imported DLLS and functions an export table listing functions exported by the code, and debug symbols.
        - Windows 바이너리는 DOS 호환 헤더, PE 헤더, 프로그램 코드가 포함된 텍스트 섹션, 초기화된 데이터가 포함된 데이터 섹션, 가져온 DLL 및 함수가 나열된 Import Table 및 코드에서 내보낸 함수가 나열된 Export Table, 마지막으로 디버그 심볼로 구성됩니다.

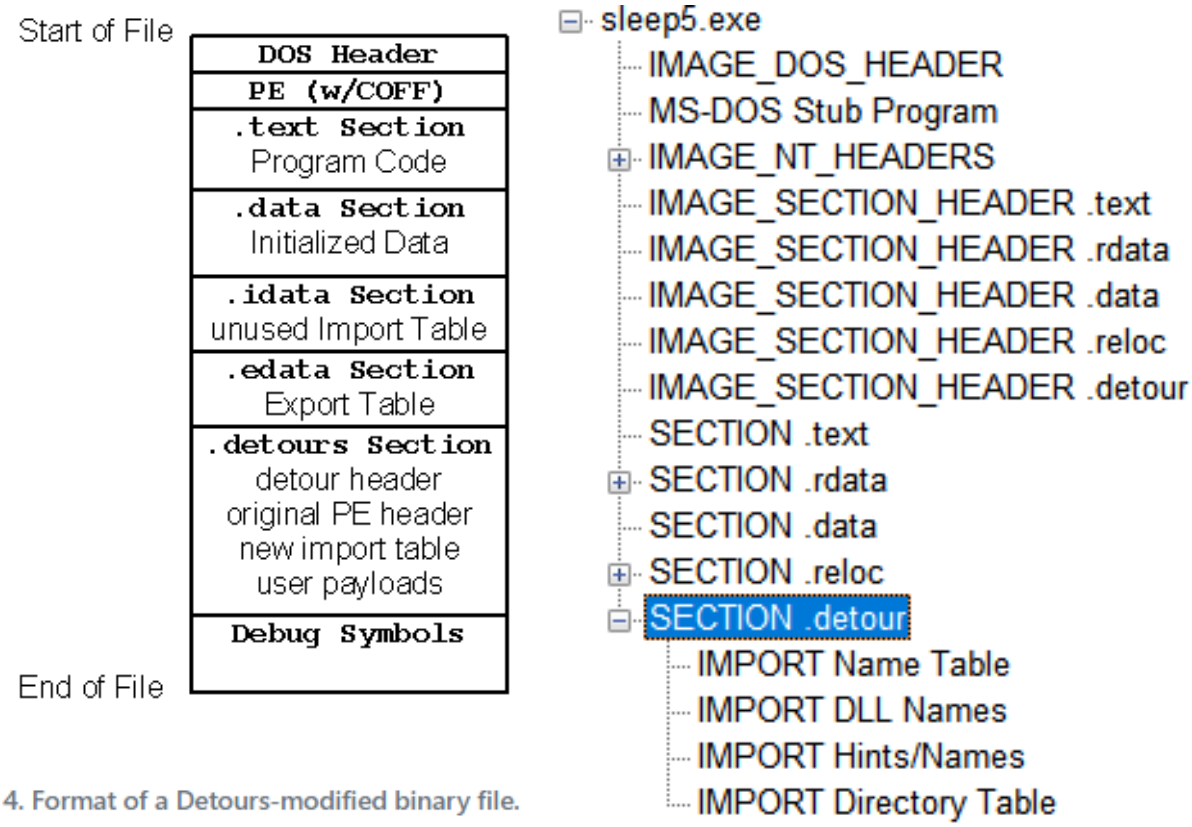


Figure 4. Format of a Detours-modified binary file.

- To modify a Windows binary, Detours creates a new '.detours' section between the export table and the debug symbols, as shown in Figure 4.
  - Windows 바이너리르 수정하기 위해 Detuours는 그림 4와 같이 edata Section과 Debug Symbols 사이에 새로운 '.detours' 섹션을 만듭니다.
  - Note that debug symbols must always reside last in a Windows binary.
    - 디버그 심볼은 항상 Windows 바이너리에서 마지막에 있어야 한다.
  - The new section contains a detours header record and a copy of the original PE header.
    - 새 섹션에는 detour 헤더 레코드와 원래 PE 헤더의 사본이 포함되어 있습니다.
- If modifying the import table, Detours creates the new import table, appends it to the copied PE header, the modifies the original PE header to point to the new import table.
  - 만일 IT를 수정하면, Detours는 새로운 IT를 작성하여 복사된 PE 헤더에 추가한 다음 원본 PE 헤더가 새롭게 생성된 IT를 가리키도록 합니다.
- Finally, Detours writes any user payloads at the end of the '.detours' section and appends the debug symbols to finish the file
  - 마지막으로 Detours는 '.detour' 섹션을 끝에 사용자 페이로드를 작성하고 디버그 심볼을 추가하여 파일을 완료합니다.

- Detours can revers modifications to the Windows binary by restoring the original PE header from the '.detours' section and removing the '.detours' section.
  - Detours는 '.detours' 섹션에서 원래 PE 헤더를 복원하고 '.detours' 섹션을 제거하여 Windows 바이너리 수정을 되돌릴 수 있습니다.
- Creating a new import table serves two purposes.
  - 새 IT는 2가지 목적으로 생성됩니다.
  - First, it preserves the original import table in case the programmer needs to reverse all modifications to the Windows file.
    - 먼저 프로그래머가 Windows 파일의 모든 수정사항을 되돌려야 할 경우를 대비하여 원본 Import Table을 유지하기 위해서다.
  - Second, the new import table can contain renamed import DLLs and functions or entirely new DLLs and functions.
    - 둘째, 새로운 Import table에는 이름이 변경된 Import DLL 및 함수 또는 완전히 새로운 DLL 및 함수로 포함될 수 있습니다.
  - For example, the setdll.exe program included in the Detours package, inserts an initial entry for a user's DLL into a target application binary.
    - 예를 들어 Detours 패키지에 포함된 setdll.exe 프로그램은 사용자 DLL의 초기 항목을 대상 응용 프로그램 바이너리에 삽입합니다.
    - As the first entry in the application's import table, the user's DLL is walways the first DLL to run in the application's address spcae.
      - 응용 프로그램 Import table의 첫 번째 항목인 사용자의 DLL은 항상 응용프로그램의 주소 공간에서 실행되는 첫 번째 DLL입니다.
- Detours provides APIs for editing import tables(DetourBinaryEditImports), adding payloads(DetourBinarySetPayload), enumerating payloads(DetourBinaryEnmeratePayloads), and removing payloads(DetourBinaryPurgePayloads).
  - Detours는 Import tables 편집, 페이로드 추가, 열거, 제거를 위한 API를 제공합니다.
- Detours also provides APIs for enumerating the binary files mapped into an address space(DetoursEnumerateModules) and locating payloads within those.
  - 또한 Detours는 주소 공간에 매핑된 이진 파일을 열거하고 매핑된 이진내에서 페이로드를 찾는 API를 제공합니다.
  - Each payload is identified by a 128-bit globally unique identifier(GUID).
    - GUID; Globally Unique Identifier
    - 각 페이로드는 128-bit GUID로 식별됩니다.
  - Payloads can be used to attach per-application configuration data to application binaries.

        - 페이로드를 사용하여 어플리케이션 별 구성 데이터를 어플리케이션 바이너리에 첨부 할 수 있습니다.
- Payloads can be copied directly into a target process using the DetourCopyPayloadToProcess API.
- 페이로드는 DetourCopyPayloadToProcess API를 사용하여 대상 프로세스로 직접 복사 할 수 있습니다.

4. Detouring 32-bit and 64-bit Processes

- The most common usage scenario for Detours is to detour function in an existing application without modifying the original application binaries.
    - Detours의 가장 일반적인 사용 시나리오는 원본 응용 프로그램 바이너리를 수정하지 않고 기존 응용 프로그램의 기능을 우회하는 것입니다.
    - In these scenarios, the user-supplied detour functions are packaged in a DLL that is loaded into the application at startup time using the DetourCreateProcessWithDll API.
        - 이러한 시나리오에서 사용자 제공 우회 함수는 시작시 DetourCreateProcessWithDll API를 사용하여 응용 프로그램에 로드되는 DLL에 패키지됩니다.
    - The DetourCreateProcessWithDLL API is called from the parent Process;
        - It alters the in-memory copy of the application by inserting an import table entry for the detour DLL.
            - Detour DLL에 대한 Import table 항목을 삽입하여 응용프로그램의 메모리 내 사본을 변경합니다.
        - This new Import table entry causes the OS loader to load the DLL after the application process has started, but before any of the application code can run.
            - 이 새로운 Import table 항목으로 인해 응용프로그램 프로세스가 시작된 후 응용 프로그램 코드가 실행되기 전에 OS 로더가 DLL을 로드합니다.
            - The detour DLL then has a chance to hook target functions in the target process.
                - Detour DLL은 이때 타겟 프로세스에서 타겟 기능을 연결할 수 있습니다.
- in computers with 64-bit processors, Windows supports both 32-bit and 64-bit applications.
    - 64-bit 프로세서가 있는 컴퓨터에서 Windows는 32-bit와 64-bit 응용 프로그램을 모두 지원합니다.
    - To support both 32-bit and 64-bit applications, you must create both 32-bitr and 64-bit versions of your detour DLL.

        - 32-bit및 64-bit 버전의 DLL을 모두 만들어야 한다.
        - You must also replace all uses of the DetourCreateProcessWithDll API with either the DetourCreateProcessWithDllEx or DetourCreateProcessWithDlls API.
            - 또한 DetourCreateProcessWithDll API의 모든 사용을 DetourCreateProcessWithDllEx 또는 DetourCreateProcessWithDlls API로 바꿔야 합니다.
            - The DetourCreateProcessWithDllEx and DetourCreateProcessWithDlls APIs choose between the 32-bit or 64-bit versions of your DLL as appropriate for the target application.
                - 각 API는 타겟 응용프로그램에 적합한 DLL의 32-bit 또는 64-bit 버전 중에서 선택합니다.

[What To do]

- Export the DetourFinishHelperProcess API as export ordinal 1.
    - DetourFinishHelperProcess API를 내보내기 서수 1로 내 보냅니다.
    - Call the DetourIsHelperProcess API in your 'DllMain' function.
        - DllMain 함수에서 DetourIsHelperProcess API를 호출하십시오.
            - Immediately return TRUE if DetourIsHelperProcess return TRUE.
                - DetourIsHelperProcess가 TRUE를 반환하면 즉시 TRUE를 반환합니다.
    - Call the DeCreateProcessWithDllEx or DetourCreateProcessWithDlls API instead of DetourCreateProcessWithDll to create new Target Process
        - 새 타겟 프로세스를 작성하려면 DetourCreateProcessWithDll 대신 DetourCreateProcessWithDllEx 또는 DetourCreateProcessWithDlls API를 호출

[How It Works]

- In the case where both the parent process and the target process are the same, such a both 32-bit or both 64-bit, the DetourCreateProcessWithDllEx API works the same as the DetourCreateProcessWithDll API.
    - 부모 프로세스와 대상 프로세스가 모두 32 비트 또는 64 비트와 같은 경우 DetourCreateProcessWithDllEx API는 DetourCreateProcessWithDll API와 동일하게 작동합니다.
- When the parent process is 32-bit and the target is 64-bit or the parent is 64-bit and the target is 32-bit DetourCreateProcessWithDllEx creates a helper process by loading your DLL into a rundll32.exe process and calling DetourFinishHelperProcess through export Ordinal 1.
    - 작성한 환경과 작성된 프로세스가 서로 다를 경우 DetourCreateProcessWithDllEx는 사용자가 생성한 Dll을 rundll32.exe 프로세스를 호출하여 로드하고 Export

Ordinal 1 함수를 통해 DetourFinishHelperProcess를 호출하여 도우미 프로세스를 만듭니다.

- This API Patches up the application's import table using the correct 32-bit or 64-bit code.
  - 이 API는 올바른 32bit 또는 64bit 코드를 사용하여 응용프로그램의 Import Table을 패치합니다.

참고

[Blog]

https://wendys.tistory.com/category/DEVELOPMENT/WIndows%20Hooking

'DEVELOPMENT/WIndows Hooking' 카테고리의 글 목록 •

https://wendys.tistory.com/161?category=757973

[C++] 최고의 API Hooking Library MS Detours. • wendys.tistory.com

https://secmem.tistory.com/480

[6기 대구 허정욱] Detours 3.0 Express #1.API Hooking • secmem.tistory.com

[Book]

- 해킹, 파괴의 광학