

```
1 // HookTester.cpp : Defines the entry point for the console application.
2 //
3
4 #include "stdafx.h"
5 #include <windows.h>
6
7 int main(int argc, char* argv)
8 {
9
10     MessageBox(
11         NULL,
12         (LPCWSTR)L"Sleeping for 3 seconds...",
13         (LPCWSTR)L"FAILURE",
14         MB_ICONASTERISK | MB_OK
15     );
16
17     Sleep(3000);
18
19     MessageBox(
20         NULL,
21         (LPCWSTR)L"Done sleeping.",
22         (LPCWSTR)L"FAILURE",
23         MB_ICONASTERISK | MB_OK
24     );
25
26     HANDLE hFile;
27     char DataBuffer[] = "This is some test data to write to the file.";
28     DWORD dwBytesToWrite = (DWORD)strlen(DataBuffer);
29     DWORD dwBytesWritten = 0;
30     BOOL bErrorFlag = FALSE;
31
32     hFile = CreateFile((LPCWSTR)L"C:\\Users\\John Smith\\Desktop\\
33         \\Testfile.txt", // name of the write
34         GENERIC_WRITE, // open for writing
35         0, // do not share
36         NULL, // default security
37         CREATE_NEW, // create new file only
38         FILE_ATTRIBUTE_NORMAL, // normal file
39         NULL); // no attr. template
40
41     WriteFile(hFile, // open file handle
42         DataBuffer, // start of data to write
43         dwBytesToWrite, // number of bytes to write
44         &dwBytesWritten, // number of bytes that were written
45         NULL); // no overlapped structure
46
47     CloseHandle(hFile);
48
49     return 0;
50 }
51
```

```
1  // Lab_07_B.cpp : Defines the entry point for the console application.
2  //
3
4  #include "stdafx.h"
5  #include <iostream>
6  #include <windows.h>
7  #include <tlhelp32.h>
8  #include <cstdio>
9
10 int main(int argc, char* argv[])
11 {
12     if (argc != 3)
13     {
14         std::cout << "Usage: Lab_07_B Evil.dll <PID>" << std::endl;
15     }
16
17     PROCESSENTRY32 entry;
18     entry.dwSize = sizeof(PROCESSENTRY32);
19
20     HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, NULL);
21
22     if (Process32First(snapshot, &entry) == TRUE)
23     {
24         while (Process32Next(snapshot, &entry) == TRUE)
25         {
26             size_t i = 0;
27             if (entry.th32ProcessID == std::atoi(argv[2]))
28             {
29                 HANDLE hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE,
30                     entry.th32ProcessID);
31                 HANDLE hThread;
32                 char* szLibPath = argv[1];
33                 void* pLibRemote; // The address (in the remote process) where
34                                     // szLibPath will be copied to;
35                 DWORD hLibModule; // Base address of loaded module
36                                     // (==HMODULE);
37                 HMODULE hKernel32 = GetModuleHandle(L"Kernel32");
38
39                 // 1. Allocate memory in the remote process for szLibPath
40                 // 2. Write szLibPath to the allocated memory
41                 pLibRemote = VirtualAllocEx(hProcess, NULL, strlen(szLibPath),
42                     MEM_COMMIT, PAGE_READWRITE);
43                 WriteProcessMemory(hProcess, pLibRemote, (void*)szLibPath, strlen
44                     (szLibPath), NULL);
45
46                 // Load "Evil.dll" into the remote process
47                 hThread = CreateRemoteThread(hProcess, NULL, 0,
48                     (LPTHREAD_START_ROUTINE)GetProcAddress(hKernel32,
49                         "LoadLibraryA"), pLibRemote, 0, NULL);
50
51                 WaitForSingleObject(hThread, INFINITE);
52
53                 // Get handle of the loaded module
```

```
46         GetExitCodeThread(hThread, &hLibModule);
47
48         // Clean up
49         CloseHandle(hThread);
50         VirtualFreeEx(hProcess, pLibRemote, sizeof(szLibPath),
51             MEM_RELEASE);
52     }
53 }
54
55 CloseHandle(snapshot);
56
57 return 0;
58 }
59
60
```

```
1  #pragma comment(lib, "detours.lib")
2
3  #include <stdio.h>
4  #include <windows.h>
5  #include "detours.h"
6
7  static VOID(WINAPI * TrueSleep)(DWORD dwMilliseconds) = Sleep;
8  static INT(WINAPI * TrueMessageBox)(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, ↗
    UINT uType) = MessageBox;
9  static HANDLE(WINAPI * TrueCreateFile)(LPCTSTR lpFileName,
10                                         DWORD dwDesiredAccess,
11                                         DWORD dwShareMode,
12                                         LPSECURITY_ATTRIBUTES lpSecurityAttributes,
13                                         DWORD dwCreationDisposition,
14                                         DWORD dwFlagsAndAttributes,
15                                         HANDLE hTemplateFile) = CreateFile;
16
17 VOID WINAPI NoSleep(DWORD dwMilliseconds)
18 {
19     printf("Program attempted to sleep for %d milliseconds.\n", dwMilliseconds);
20     return TrueSleep(0);
21 }
22
23 INT WINAPI ChangeMessageBoxTitle(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, ↗
    UINT uType)
24 {
25     LPCTSTR new_title = L"SUCCESS!";
26     wprintf(L"Changing MessageBox title from %s to %s\n", lpCaption, new_title);
27     return TrueMessageBox(hWnd, lpText, new_title, uType);
28 }
29
30 HANDLE WINAPI LogFileCreation(LPCTSTR lpFileName,
31                                DWORD dwDesiredAccess,
32                                DWORD dwShareMode,
33                                LPSECURITY_ATTRIBUTES lpSecurityAttributes,
34                                DWORD dwCreationDisposition,
35                                DWORD dwFlagsAndAttributes,
36                                HANDLE hTemplateFile)
37 {
38     wprintf(L"Program attempted to create file %s\n", lpFileName);
39     return TrueCreateFile(lpFileName,
40                            dwDesiredAccess,
41                            dwShareMode,
42                            lpSecurityAttributes,
43                            dwCreationDisposition,
44                            dwFlagsAndAttributes,
45                            hTemplateFile);
46 }
47
48 BOOL WINAPI DllMain(HINSTANCE hinst, DWORD dwReason, LPVOID reserved)
49 {
50     LONG error;
```

```
51     (void)hinst;
52     (void)reserved;
53
54     if (DetourIsHelperProcess()) {
55         return TRUE;
56     }
57
58     if (dwReason == DLL_PROCESS_ATTACH) {
59         DetourRestoreAfterWith();
60
61         printf("Starting.\n");
62         fflush(stdout);
63
64         DetourTransactionBegin();
65         DetourUpdateThread(GetCurrentThread());
66         DetourAttach(&(PVOID&)TrueSleep, NoSleep);
67         error = DetourTransactionCommit();
68
69         if (error == NO_ERROR) {
70             printf("Detoured Sleep().\n");
71         }
72         else {
73             printf("Error detouring Sleep(): %d\n", error);
74         }
75
76         DetourTransactionBegin();
77         DetourUpdateThread(GetCurrentThread());
78         DetourAttach(&(PVOID&)TrueMessageBox, ChangeMessageBoxTitle);
79         error = DetourTransactionCommit();
80
81         if (error == NO_ERROR) {
82             printf("Detoured MessageBox().\n");
83         }
84         else {
85             printf("Error detouring MessageBox(): %d\n", error);
86         }
87
88         DetourTransactionBegin();
89         DetourUpdateThread(GetCurrentThread());
90         DetourAttach(&(PVOID&)TrueCreateFile, LogFileCreation);
91         error = DetourTransactionCommit();
92
93         if (error == NO_ERROR) {
94             printf("Detoured CreateFile().\n");
95         }
96         else {
97             printf("Error detouring CreateFile(): %d\n", error);
98         }
99     }
100 }
101 else if (dwReason == DLL_PROCESS_DETACH) {
102     DetourTransactionBegin();
```

```
103     DetourUpdateThread(GetCurrentThread());
104     DetourDetach(&(PVOID&)TrueSleep, NoSleep);
105     error = DetourTransactionCommit();
106     printf("Removed Sleep() (result=%d)\n", error);
107     fflush(stdout);
108
109     DetourTransactionBegin();
110     DetourUpdateThread(GetCurrentThread());
111     DetourDetach(&(PVOID&)TrueMessageBox, ChangeMessageBoxTitle);
112     error = DetourTransactionCommit();
113     printf("Removed MessageBox() (result=%d)\n", error);
114     fflush(stdout);
115
116     DetourTransactionBegin();
117     DetourUpdateThread(GetCurrentThread());
118     DetourDetach(&(PVOID&)TrueCreateFile, LogFileCreation);
119     error = DetourTransactionCommit();
120     printf("Removed CreateFile() (result=%d)", error);
121     fflush(stdout);
122 }
123
124     return TRUE;
125 }
126
127 extern "C" __declspec(dllexport) void dummy(void) {
128     return;
129 }
130
```