```
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,
            (LPCWSTR)L"Sleeping for 3 seconds...",
12
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;
        char DataBuffer[] = "This is some test data to write to the file.";
27
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\
                                                                                        P
                                         // name of the write
          \Testfile.txt",
33
                            GENERIC WRITE,
                                                    // open for writing
34
                            0,
                                                    // do not share
35
                            NULL,
                                                    // default security
36
                            CREATE_NEW,
                                                    // create new file only
37
                            FILE_ATTRIBUTE_NORMAL, // normal file
                                                    // no attr. template
38
                            NULL);
39
40
                                    // open file handle
         WriteFile(hFile,
41
                                     // start of data to write
                    DataBuffer,
42
                    dwBytesToWrite, // number of bytes to write
                    &dwBytesWritten, // number of bytes that were written
43
44
                    NULL);
                                      // no overlapped structure
45
46
        CloseHandle(hFile);
47
48
       return 0;
49 }
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>
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;</pre>
15
16
17
       PROCESSENTRY32 entry;
18
        entry.dwSize = sizeof(PROCESSENTRY32);
19
20
       HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS SNAPPROCESS, NULL);
21
       if (Process32First(snapshot, &entry) == TRUE)
22
23
           while (Process32Next(snapshot, &entry) == TRUE)
24
25
            {
26
                size_t i = 0;
27
                if (entry.th32ProcessID == std::atoi(argv[2]))
28
29
                    HANDLE hProcess = OpenProcess(PROCESS ALL ACCESS, FALSE,
                      entry.th32ProcessID);
30
                    HANDLE hThread;
31
                            szLibPath = argv[1];
                    char*
32
                    void*
                            pLibRemote; // The address (in the remote process) where →
                       szLibPath will be copied to;
33
                    DWORD
                           hLibModule; // Base address of loaded module
                      (==HMODULE);
34
                    HMODULE hKernel32 = GetModuleHandle(L"Kernel32");
35
36
                    // 1. Allocate memory in the remote process for szLibPath
37
                    // 2. Write szLibPath to the allocated memory
38
                    pLibRemote = VirtualAllocEx(hProcess, NULL, strlen(szLibPath),
                      MEM_COMMIT, PAGE_READWRITE);
39
                    WriteProcessMemory(hProcess, pLibRemote, (void*)szLibPath, strlen →
                      (szLibPath), NULL);
40
                    // Load "Evil.dll" into the remote process
41
42
                    hThread = CreateRemoteThread(hProcess, NULL, 0,
                      (LPTHREAD_START_ROUTINE)GetProcAddress(hKernel32,
                      "LoadLibraryA"), pLibRemote, 0, NULL);
43
                    WaitForSingleObject(hThread, INFINITE);
44
45
                    // Get handle of the loaded module
```

```
c:\Users\John Smith\Desktop\Lab_07_B\Lab_07_B\Lab_07_B.cpp
                                                                                       2
46
                   GetExitCodeThread(hThread, &hLibModule);
47
                   // Clean up
48
49
                   CloseHandle(hThread);
                   VirtualFreeEx(hProcess, pLibRemote, sizeof(szLibPath),
50
                     MEM_RELEASE);
51
                }
           }
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 }
23 INT WINAPI ChangeMessageBoxTitle(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption,
     UINT uType)
24 {
       LPCTSTR new title = L"SUCCESS!";
25
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,
           hTemplateFile);
45
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
             printf("Starting.\n");
61
             fflush(stdout);
62
63
64
             DetourTransactionBegin();
65
             DetourUpdateThread(GetCurrentThread());
66
             DetourAttach(&(PVOID&)TrueSleep, NoSleep);
67
             error = DetourTransactionCommit();
68
             if (error == NO_ERROR) {
69
 70
                 printf("Detoured Sleep().\n");
71
             }
            else {
72
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 {
                 printf("Error detouring CreateFile(): %d\n", error);
97
98
             }
99
100
        }
101
        else if (dwReason == DLL PROCESS DETACH) {
102
             DetourTransactionBegin();
```

```
C:\Users\John Smith\Desktop\Lab_07_A\Lab_07\dllmain.cpp
```

```
3
```

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