

Forum:[Hack Forums](#)

Thread: [\[Whats wrong here?\] Thread hijacking SIR](#)

SIM-3466371 - January 5, 2019, 8:18 am

I have a simply process called Loader.exe

```
Code #include <windows.h>
#include <iostream>
```

```
typedef int(__stdcall *f_func)();
```

```
// Loads a dll and call method abc, in this dll there are other method, called "secret"
// this process finishes very fast, reason why I believe that it will be necessary to throw it suspended
state
```

```
int main()
```

```
{
    HINSTANCE hGetProcIDDLL =
    LoadLibrary("C:\\Users\\Admin\\source\\repos\\CodeCave\\x64\\Release\\Dll1.dll");
```

```
    if (!hGetProcIDDLL) return EXIT_FAILURE;
```

```
    // resolve function address here
```

```
    f_func func = (f_func)GetProcAddress(hGetProcIDDLL, "abd");
    if (!func) return EXIT_FAILURE;
```

```
    func();// Call dummy function
```

```
    return EXIT_SUCCESS;
```

```
}
```

I have a simply dll, called Dll1.dll

```
Code #include "stdafx.h"
```

```
#include <iostream>
```

```
#include <fstream>
```

```
#define EXTERN_DLL_EXPORT extern "C" __declspec(dllexport)
```

```
// Dummy
```

```
EXTERN_DLL_EXPORT int abd() {
    return 987;
}
```

```
// Dll hidden function that is never called from Loader.exe
```

```
EXTERN_DLL_EXPORT int secret() {
    MessageBox(
        NULL,
        (LPCWSTR)L"WIN",
        (LPCWSTR)L"Secret unlocked",
        MB_ICONINFORMATION | MB_CANCELTRYCONTINUE | MB_DEFBUTTON2
    );
    std::ofstream myfile;
```

```

myfile.open("C:\\Users\\Admin\\Desktop\\SECRET.txt");
myfile << "Here you have your secret\n";
myfile.close();
return 987;
}

```

```

BOOL APIENTRY DllMain( HMODULE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved
                      )
{
    std::ofstream myfile;
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

```

Now, i want to create a new process, called SIR.exe, that will launch Loader.exe and inject code in Loader.exe in order to force to execute secret function of DLL1, but: I don't want to modify the code of Loader.exe in order to execute system pause and have more time to inject code, i think i must start Loader.exe in suspended state

I want to use SIR technique (Suspend, Injects, Resume) in order to hijack the main thread

SIR.exe code:

```

Code #include <Windows.h>
#include <stdio.h>
#include <tlhelp32.h>
#include <tchar.h>
#include <memory>

```

```

// This is a pointer to our exported function in dll
typedef int (WINAPI* ExportedFunction)();

```

```

// Struct for store data[included parameters] about function exported in dll
struct ExportedFunctionData {
    uintptr_t uFuncAddress;
};

```

```

// This function obtains pri
BOOL SetSePrivilege() {
    TOKEN_PRIVILEGES tp = { 0 };
    HANDLE hToken = NULL;

```

```

    if (OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES |
        TOKEN_QUERY, &hToken)) {
        tp.PrivilegeCount = 1;

```

```

tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

if (LookupPrivilegeValue(NULL, SE_DEBUG_NAME, &tp.Privileges[0].Luid)) {
    if (AdjustTokenPrivileges(hToken, FALSE, &tp, 0, NULL, NULL) == 0) {
        wprintf(L"[!] Error: AdjustTokenPrivilege failed! %u\n", GetLastError());

        if (GetLastError() == ERROR_NOT_ALL_ASSIGNED)
        {
            wprintf(L"[*] Warning: The token does not have the specified privilege.\n");
            return FALSE;
        }
    }
}

CloseHandle(hToken);
}
else
    return FALSE;

return TRUE;
}

/*
 * Function that will call to our function
 */
DWORD WINAPI FunctionA(LPVOID lpParameter)
{
    auto data = reinterpret_cast<ExportedFunctionData*>(lpParameter);
    auto func = reinterpret_cast<ExportedFunction>(data->uFuncAddress);

    func();

    return EXIT_SUCCESS;
}

DWORD SIR() {
    SetSePrivilege();

    // Get function pointer
    auto data = std::make_unique<ExportedFunctionData>();
    auto dDll1 = LoadLibraryA("C:\\Users\\Admin\\source\\repos\\CodeCave\\x64\\Release\\Dll1.dll");
    auto secretAddress = GetProcAddress(dDll1, "secret");

    data->uFuncAddress = reinterpret_cast<uintptr_t>(secretAddress); // Set our function in struct
    FreeLibrary(dDll1);

    // Launch process in suspended state
    PROCESS_INFORMATION proInf;
    STARTUPINFO starInf;
    ZeroMemory(&starInf, sizeof(starInf));
    ZeroMemory(&proInf, sizeof(proInf));
    starInf.cb = sizeof(starInf);

```

```

CreateProcessW(L"C:\\Users\\Admin\\source\\repos\\CodeCave\\x64\\Release\\Loader.exe",
NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &starInf, &proInf);

// Get PID and TID
DWORD PID = proInf.dwProcessId;
DWORD TID = proInf.dwThreadId;

// Handle to process and thread
HANDLE hThread = proInf.hThread;
HANDLE hProcess = proInf.hProcess;

// Allocate Space
// Here i'm using 150 bytes as size, i need to know how obtain the size of function
[FunctionA]???
auto pShellcode = VirtualAllocEx(hProcess, 0, 150, MEM_COMMIT | MEM_RESERVE,
PAGE_EXECUTE_READWRITE);
auto pDataAddress = VirtualAllocEx(hProcess, 0, sizeof(*data.get()), MEM_COMMIT |
MEM_RESERVE, PAGE_EXECUTE_READWRITE);

// Write memory
WriteProcessMemory(hProcess, pShellcode, &FunctionA, 150, nullptr); // Writting code in
memory
WriteProcessMemory(hProcess, pDataAddress, data.get(), sizeof(*data.get()), nullptr); // Writting
params in memory

// Hijack current thread
CONTEXT ctx;
ctx.ContextFlags = CONTEXT_CONTROL;
GetThreadContext(hThread, &ctx);
unsigned long lastIP = ctx.Rip; // Get las value of IP pointer, in this case this code is only for 64
bits[Rip]
ctx.Rip = (DWORD)pShellcode; // New value for execute
ctx.ContextFlags = CONTEXT_CONTROL;

SetThreadContext(hThread, &ctx); // Execute the thread with the Rip register pointing to our
memory

ResumeThread(hThread);

Sleep(8000);

// Free memory
VirtualFreeEx(hProcess, pDataAddress, 0, MEM_RELEASE);
VirtualFreeEx(hProcess, pShellcode, 0, MEM_RELEASE);

// Close handlers
CloseHandle(hThread);
CloseHandle(hProcess);

return TRUE;
}

```

```
int main() {  
    SIR();  
    system("pause");  
}
```

Whats is wrong with my code?

Thanks in advance.

HFDone - January 5, 2019, 11:12 am

you can do it much simple by using manual mapping technique

SIM-3466371 - January 5, 2019, 11:20 am

(Jan 5, 2019 12:12 PM)HFDone Wrote: [\[CITE\]](#) you can do it much simple by using manual mapping technique

I'm looking about manual mapping, it's similar to reflective dll injection, true?

HFDone - January 5, 2019, 11:55 am

yes you load it from memory and call the function