

Forum:[Hack Forums](#)

Thread: [Using MessageBox Parameter To Intercept Loaded Modules.](#)

HiddenEyE - July 22, 2018, 4:12 am

Hi HF C++ NOOBS.

during my c/c++ learning days i always desire to know which modules and the location of the modules on disk that was loaded in the address space of my application. that made me come up with this code below and many other codes i wrote.

i present a code which you can use for the following purpose:

1. use this code below to know the loaded modules in your application.
2. to intercept modules loaded by another application by injecting it to the application of your choice.
3. can be use to check if your dll is injected (during dll injection).
4. etc

CODE for first scenario:

Code #include <windows.h>

#include <tchar.h>

#include <tlhelp32.h>

```
int WINAPI _tWinMain(HINSTANCE ,HINSTANCE ,LPTSTR ,int)
```

```
{  
    TCHAR MsgBuffer[8192] = _T("MODULE LIST AND THEIR LOCATION:\r\n");
```

```
    HANDLE TakeSnap = CreateToolhelp32Snapshot(  
        TH32CS_SNAPMODULE, GetCurrentProcessId());  
    if(TakeSnap == INVALID_HANDLE_VALUE)  
        return -1;
```

```
    MODULEENTRY32 TheDll;  
    TheDll.dwSize = sizeof(TheDll);  
    BOOL ModuleOutput = Module32First(TakeSnap, &TheDll);  
    while(ModuleOutput) {  
        lstrcat(MsgBuffer, _T("\r\n"));  
        lstrcat(MsgBuffer, _T("\r\n"));  
        lstrcat(MsgBuffer, (LPSTR)TheDll.szModule);
```

```
        lstrcat(MsgBuffer, _T("\r\n"));  
        lstrcat(MsgBuffer, (LPSTR)TheDll.szExePath);  
        ModuleOutput = Module32Next(TakeSnap, &TheDll);
```

```
    }  
    CloseHandle(TakeSnap);  
    MessageBox(GetActiveWindow(), MsgBuffer, _T("MODULE LIST AND THEIR LOCATION"),  
    MB_OK);  
    return 0;
```

```
}  
CODE for second scenario:
```

Code #include <windows.h>

#include <tchar.h>

#include <tlhelp32.h>

```

void HiddenEyE()
{
    TCHAR MsgBuffer[8192] = _T("MODULE LIST AND THEIR LOCATION:\r\n");

    HANDLE TakeSnap = CreateToolhelp32Snapshot(
        TH32CS_SNAPMODULE, GetCurrentProcessId());
    if(TakeSnap == INVALID_HANDLE_VALUE)
    {}

    MODULEENTRY32 TheDll;
    TheDll.dwSize = sizeof(TheDll);
    BOOL ModuleOutput = Module32First(TakeSnap, &TheDll);
    while(ModuleOutput) {
        lstrcat(MsgBuffer, _T("\r\n"));
        lstrcat(MsgBuffer, _T("\r\n"));
        lstrcat(MsgBuffer, (LPSTR)TheDll.szModule);

        lstrcat(MsgBuffer, _T("\r\n"));
        lstrcat(MsgBuffer, (LPSTR)TheDll.szExePath);
        ModuleOutput = Module32Next(TakeSnap, &TheDll);
    }
    CloseHandle(TakeSnap);
    MessageBox(GetActiveWindow(), MsgBuffer, _T("MODULE LIST AND THEIR LOCATION"),
    MB_OK);
}
BOOL WINAPI DllMain(HMODULE hModule, DWORD dwReason, LPVOID lpReserved)
{
    switch (dwReason)
    {
    case DLL_PROCESS_ATTACH:
        HiddenEyE();
        break;
    case DLL_PROCESS_DETACH:
        MessageBoxW(NULL, L"DLL UNLOADED!", L"HIDDEN EYEDLL", MB_ICONINFORMATION);
        break;
    default:
        break;
    }

    return TRUE;
}

```

HOPE SOMEONE FINDS IT USEFUL

criticism are welcomed but remember it was written years ago.

0xDEAD10CC - July 22, 2018, 7:32 pm

This is the most basic method of doing it.. Another method is to walk the PEB. Common knowledge for many people that have used DLL's for plugin purposes outside of managed world where DLL's actually have an entrypoint.

vmcall - July 23, 2018, 8:10 am

This doesn't even intercept anything..

To properly intercept (natively) loaded modules, you would have to hook LoadLibrary (or any of it's native ldrapi counterparts: LdrLoadDll -> LdrpLoadDll -> LdrpLoadDllInternal ->

LdrpFindOrPrepareLoadingModule).

To properly intercept mapped modules, you would have to hook any function that get's called before RtlUserThreadStart, and comparing thread start address against module list to find out if it is 'foreign' code.

HiddenEye - July 24, 2018, 3:20 am

(Jul 22, 2018 07:32 PM)0xDEAD10CC Wrote: [\[CITE\]](#) This is the most basic method of doing it.. Another method is to walk the PEB. Common knowledge for many people that have used DLL's for plugin purposes outside of managed world where DLL's actually have an entrypoint.

thanks for posting in my thread.

yes i agree with their are many ways out there. But actually the thread was for noobs which i indicated at the beginning of the thread.

came up with the idea during my learning stage.

LOL.

(Jul 23, 2018 08:10 AM)vmcall Wrote: [\[CITE\]](#) This doesn't even intercept anything..

To properly intercept (natively) loaded modules, you would have to hook LoadLibrary (or any of it's native ldrapi counterparts: LdrLoadDll -> LdrpLoadDll -> LdrpLoadDllInternal ->

LdrpFindOrPrepareLoadingModule).

To properly intercept mapped modules, you would have to hook any function that get's called before RtlUserThreadStart, and comparing thread start address against module list to find out if it is 'foreign' code.

thanks for the post.

actually you are right, but the code above was just intended for noobs new to c/c++

and i wrote that during my beginner stage in c/c++.

anyway you should not expect a noob in c/c++ to be playing with hooking when they need to understand functions(winAPI FUNCTIONS) , its parameters and how to manipulate them for good use..