

Forum:[Hack Forums](#)

Thread: [Detours : Hook function from its address?](#)

Wozniak - May 3, 2015, 9:14 am

I tried doing it without using detours but it is a lot more coding, and I don't want to worry about doing something wrong, so just for testing I wanted to use MSDetours.

I have tried tuts/examples from multiple sites.

Program to Hook:

PHP Code

```
<?php
#include <iostream>
#include <Windows.h>

int hookMe(int nput)
{
    std::cout << "This is the hooked function" << std::endl;
    std::cout << nput << std::endl;
    return nput;
}

int main()
{
    hookMe(1);
    Sleep(5000);
    hookMe(1);
    Sleep(5000);
    hookMe(1);
    return 0;
}
```

IDA Pro Address: (I think this changes, am I getting it wrong?)

<http://prntscr.com/712fl2>

The DLL is auto-injected, result:

<http://prntscr.com/712iuw>

Nothing changed.

But injection is successful:

<http://prntscr.com/712m42>

HookMsg (dll) code:

PHP Code

```
<?php
#include <iostream>
#include <windows.h>
#include <detours.h>

#pragma comment(lib, "detours")
```

```

#define ADDRESS 0x00412C30 //This is the address where our targeted function begins
int (__cdecl* originalFunction)(int);
//Pointer to the function we want to hook, must be declared same as original

int hookedFunction(int hookedInput)
{
    std::cout << "original function: argument = " << hookedInput << std::endl;
    //we can access arguments passed to original function
    hookedInput = 50;    //Modify arguments
    return originalFunction(hookedInput);
}

INT APIENTRY DllMain(HMODULE hDLL, DWORD Reason, LPVOID Reserved)
{
    switch (Reason)
    {
        case DLL_PROCESS_ATTACH:
            OutputDebugString(L"injected successfully");
            DisableThreadLibraryCalls(hDLL);
            DetourTransactionBegin();
            DetourUpdateThread(GetCurrentThread());
            DetourAttach(&(PVOID&)originalFunction, hookedFunction);
            if (DetourTransactionCommit() == NO_ERROR)
                OutputDebugString(L"detoured successfully");
            break;
        case DLL_PROCESS_DETACH:
            break;
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
            break;
    }
    return TRUE;
}

```

So, is it not working because I have the wrong address, or did I do it wrong way more?
(How can I check what error I got from detours, maybe thats helpful)

Edit:

I got it to work, but only using a known function like CreateFile.

What I need is to hook a function from its address obtained from IDA.

Prsprs - May 3, 2015, 12:55 pm

You should not really do this:

Code #define ADDRESS 0x00412C30 //This is the address where our targeted function begins

Instead get the offset relative to the image base and then, from your dll, do something like this:

Code #define _FUNCTIONOFFSET_ 0x12C30 // May no be the one tho, just an example

.
.

.

Once injected, in one of your functions:

Code DWORD dwBaseAddress = (DWORD)GetModuleHandle(NULL); // DWORD64 / QWORD if 64bit application

DWORD dwTargetFunction = (dwBaseAddress + _FUNCTIONOFFSET_);
Then, what i'd do is a trampoline hook, without using MSDetour or any external library

You'd need to use VirtualProtect to change the page protection so you can change the bytes on the .text section.

Change the bytes making it a jump to your function (the hook function)
and make sure the hook function doesn't corrupt the stack, (_asm PUSHAD and _asm POPAD would do that) also make sure it gets back to the original function.

Remix - May 3, 2015, 1:33 pm

Here's a little example code using the detours.h lib by microsoft:

Code #include <detours.h>

// function prototype: int sub_5(const char*)

typedef int(*hook)(const char*) ;
hook func; // original function

```
int hookedFunc(const char* a) {  
    return func("hooked");  
}
```

```
BOOL APIENTRY DllMain( HMODULE hModule,  
    DWORD ul_reason_for_call,  
    LPVOID lpReserved  
    )  
{  
    switch (ul_reason_for_call)  
    {  
        case DLL_PROCESS_ATTACH:  
            func = (hook) DetourFunction((PBYTE) 0x003451D6, (PBYTE) hookedFunc);  
        case DLL_THREAD_ATTACH:  
        case DLL_THREAD_DETACH:  
        case DLL_PROCESS_DETACH:  
            break;  
    }  
    return TRUE;  
}
```

8086 - May 3, 2015, 2:20 pm

(May 3, 2015 12:55 PM)Prsprs Wrote: [\[CITE\]](#) You should not really do this:

Code #define ADDRESS 0x00412C30 //This is the address where our targeted function begins
Instead get the offset relative to the image base and then, from your dll, do something like this:

Code #define _FUNCTIONOFFSET_ 0x12C30 // May no be the one tho, just an example

.
. .
.

Once injected, in one of your functions:

Code DWORD dwBaseAddress = (DWORD)GetModuleHandle(NULL); // DWORD64 / QWORD if 64bit application

DWORD dwTargetFunction = (dwBaseAddress + _FUNCTIONOFFSET_);

Then, what i'd do is a trampoline hook, without using MSDetour or any external library

You'd need to use VirtualProtect to change the page protection so you can change the bytes on the .text section.

Change the bytes making it a jump to your function (the hook function)

and make sure the hook function doesn't corrupt the stack, (_asm PUSHAD and _asm POPAD would do that) also make sure it gets back to the original function.

Are you retarded?

You realize he never said this was 64 bit.

Besides if it was 64 bit, you wouldn't be able to do pushad and popad because those instructions don't exist in the 64 bit instruction set. Aside from that, you're not even allowed to use inline assembly in 64 bit.

MS Detours is a trampoline hook.

MS Detours takes care of the page protection.

MS Detours takes care of the trampoline.

MS Detours takes care of restoring the original function.

MS Detours doesn't even support 64 bit! Unless you buy the professional version for 10.000! So clearly he's doing this in 32 bit you brain-dead monkey.

No, pushad and popad is an awful idea because it's pointlessly using up stack space when it isn't necessary.

If you're performing jumps rather than calls, then you don't need to account for the stack pointer. You also shouldn't be using any non volatile registers in your hooks and if you are going to use them, you should push them onto the stack.

But his hook isn't even in inline assembly! So there's no need to preserve any registers! What the fuck do you think the compiler is for?

@Remix your example is in 1.5, OP is using 3.0. If I was ever going to rely on Detours though, I would just use 1.5. 3.0 tries to perform atomic operations on the hooks for thread safety. It's a bit redundant honestly.

OP: If you're going to be using MS Detours, then use 1.5.

If you want to do the hook on your own.

I won't provide you with the real code, but I'll provide you with the pseudo code.

This is of course for 32 bit.

```
Code void* hook(void* dst,void* src, unsigned long len)
```

```
{
```

```
    //Nevermind, I'm not giving anything to you HF fuckers.
```

```
}Of course you should do error checking, I just didn't give a fuck enough to place it in the pseudo code.
```

Wozniak - May 3, 2015, 2:28 pm

(May 3, 2015 01:33 PM)[iRemix] Wrote: [\[CITE\]](#) Here's a little example code using the detours.h lib by microsoft:

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

```
// function prototype: int sub_5(const char*)
```

```
typedef int(*hook)(const char*) ;
```

```
hook func; // original function
```

```

int hookedFunc(const char* a) {
    return func("hooked");
}

BOOL APIENTRY DllMain( HMODULE hModule,
    DWORD ul_reason_for_call,
    LPVOID lpReserved
    )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        func = (hook) DetourFunction((PBYTE) 0x003451D6, (PBYTE) hookedFunc);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}

```

And what is the best (easiest) way to get the functions address?

8086 - May 3, 2015, 2:41 pm

(May 3, 2015 02:28 PM)_Wozniak_ Wrote: [\[CITE\]](#) And what is the best (easiest) way to get the functions address?

Get the offset from the module base.

The module base would point to the PE headers.

+4096 would get you to the .text section.

Prsprs - May 3, 2015, 4:48 pm

(May 3, 2015 02:20 PM)8086 Wrote: [\[CITE\]](#) Are you retarded?

You realize he never said this was 64 bit.

Besides if it was 64 bit, you wouldn't be able to do pushad and popad because those instructions don't exist in the 64 bit instruction set. Aside from that, you're not even allowed to use inline assembly in 64 bit.

MS Detours is a trampoline hook.

MS Detours takes care of the page protection.

MS Detours takes care of the trampoline.

MS Detours takes care of restoring the original function.

MS Detours doesn't even support 64 bit! Unless you buy the professional version for 10.000! So clearly he's doing this in 32 bit you brain-dead monkey.

No, pushad and popad is an awful idea because it's pointlessly using up stack space when it isn't necessary.

If you're performing jumps rather than calls, then you don't need to account for the stack pointer. You also shouldn't be using any non volatile registers in your hooks and if you are going to use them, you should push them onto the stack.

But his hook isn't even in inline assembly! So there's no need to preserve any registers! What the fuck do you think the compiler is for?

@Remix your example is in 1.5, OP is using 3.0. If I was ever going to rely on Detours though, I would just use 1.5. 3.0 tries to perform atomic operations on the hooks for thread safety. It's a bit redundant honestly.

OP: If you're going to be using MS Detours, then use 1.5.

If you want to do the hook on your own.

I won't provide you with the real code, but I'll provide you with the pseudo code.

This is of course for 32 bit.

```
Code void* hook(void* dst,void* src, unsigned long len)
```

```
{
```

```
    //Nevermind, I'm not giving anything to you HF fuckers.
```

```
}Of course you should do error checking, I just didn't give a fuck enough to place it in the pseudo code.
```

Other than the PUSHAD and POPAD which I forgot to mention that it's only for 32bit, everything I said above is more than right.

and lol, I only mentioned ONCE 64bit for casting when trying to get the imagebase lol, no wonder why you're -22, kiddo

(May 3, 2015 02:28 PM)_Wozniak_ Wrote: [\[CITE\]](#) And what is the best (easiest) way to get the functions address?

With IDA, rebase the program to 0x0, after that, find the function you're willing to hook and the current address it has will be its offset

so to access it from a dll, you'd need GetModuleHandle(NULL)+Offset

8086 - May 4, 2015, 6:28 am

(May 3, 2015 04:48 PM)Prsprs Wrote: [\[CITE\]](#) Other than the PUSHAD and POPAD which I forgot to mention that it's only for 32bit, everything I said above is more than right.

and lol, I only mentioned ONCE 64bit for casting when trying to get the imagebase lol, no wonder why you're -22, kiddo

With IDA, rebase the program to 0x0, after that, find the function you're willing to hook and the current address it has will be its offset

so to access it from a dll, you'd need GetModuleHandle(NULL)+Offset

I guess you've never heard of a void* then.

Stupid kiddo. You think I give a fuck about rep?

And no, what you said is vaguely right. You couldn't even perform a hot patch without copying and pasting, dumb fuck.

I've seen enough of your stupid kid.

Quote:Change the bytes making it a jump to your function (the hook function)

No. That's not correct. But I won't even bother to tell you why it's not correct because then I'd be sharing knowledge with you.

Quote:and make sure the hook function doesn't corrupt the stack, (_asm PUSHAD and _asm POPAD would do that)

Again, that's not correct. pushad and popad will just preserve registers. It will in no way prevent you from corrupting the stack. But again, you've proven you have no idea what the fuck you're talking about.

Ponut - May 5, 2015, 5:13 am

(May 3, 2015 09:14 AM)_Wozniak_ Wrote: [CITE](#) I tried doing it without using detours but it is a lot more coding, and I don't want to worry about doing something wrong, so just for testing I wanted to use MSDetours.

I have tried tuts/examples from multiple sites.

Program to Hook:

PHP Code

```
<?php
#include <iostream>
#include <Windows.h>

int hookMe(int nput)
{
    std::cout << "This is the hooked function" << std::endl;
    std::cout << nput << std::endl;
    return nput;
}

int main()
{
    hookMe(1);
    Sleep(5000);
    hookMe(1);
    Sleep(5000);
    hookMe(1);
    return 0;
}
```

IDA Pro Address: (I think this changes, am I getting it wrong?)

<http://prntscr.com/712fl2>

The DLL is auto-injected, result:

<http://prntscr.com/712iuw>

Nothing changed.

But injection is successful:

<http://prntscr.com/712m42>

HookMsg (dll) code:

PHP Code

```
<?php
#include <iostream>
#include <windows.h>
#include <detours.h>

#pragma comment(lib, "detours")

#define ADDRESS 0x00412C30 //This is the address where our targeted function begins
int (__cdecl* originalFunction)(int);
//Pointer to the function we want to hook, must be declared same as original

int hookedFunction(int hookedInput)
```

```

{
    std::cout << "original function: argument = " << hookedInput << std::endl;
    //we can access arguments passed to original function
    hookedInput = 50;    //Modify arguments
    return originalFunction(hookedInput);
}

INT APIENTRY DllMain(HMODULE hDLL, DWORD Reason, LPVOID Reserved)
{
    switch (Reason)
    {
    case DLL_PROCESS_ATTACH:
        OutputDebugString(L"injected successfully");
        DisableThreadLibraryCalls(hDLL);
        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourAttach(&(PVOID&)originalFunction, hookedFunction);
        if (DetourTransactionCommit() == NO_ERROR)
            OutputDebugString(L"detoured successfully");
        break;
    case DLL_PROCESS_DETACH:

    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
        break;
    }
    return TRUE;
}

```

So, is it not working because I have the wrong address, or did I do it wrong way more?
(How can I check what error I got from detours, maybe thats helpful)

Edit:

I got it to work, but only using a known function like CreateFile.

What I need is to hook a function from its address obtained from IDA. What could go possibly wrong with a 5 byte jmp hook?

(x32)