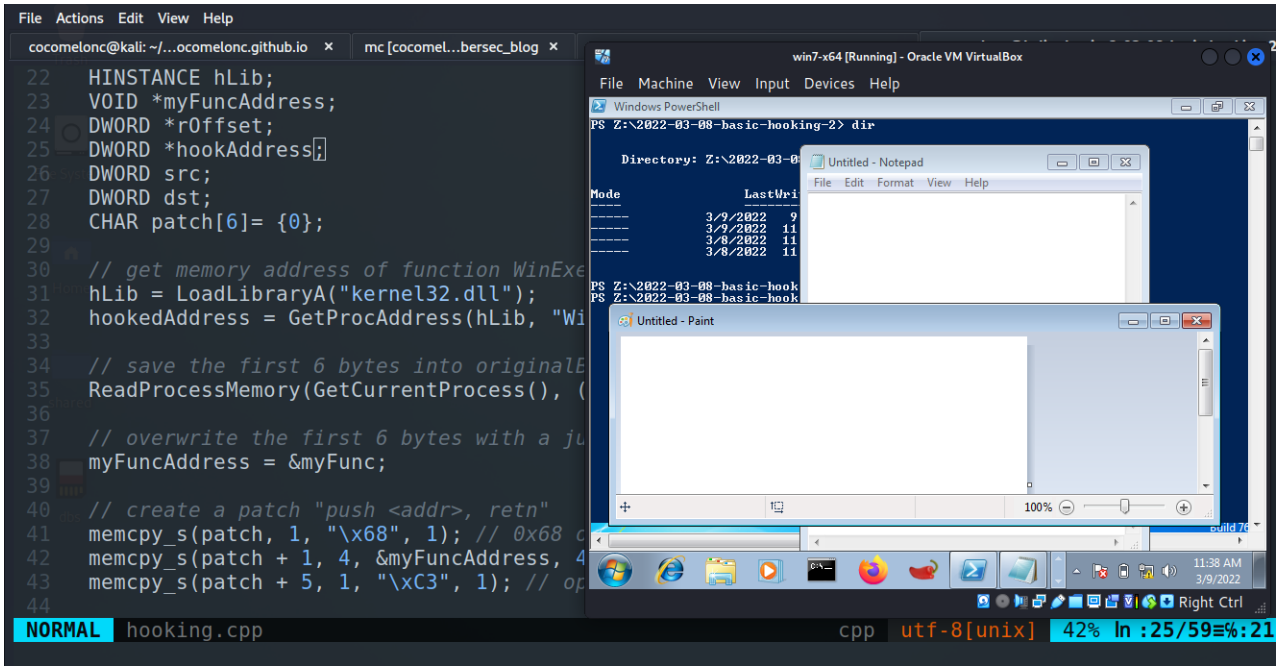
Windows API hooking part 2. Simple C++ example.

cocomelonc.github.io/tutorial/2022/03/08/basic-hooking-2.html

March 8, 2022

2 minute read

Hello, cybersecurity enthusiasts and white hackers!



what is API hooking?

API hooking is a technique by which we can instrument and modify the behaviour and flow of API calls. This technique is also used by many AV solutions to detect if code is malicious.

The easiest way of hooking is by inserting a jump instruction. In this post I will show you another technique.

This method is six bytes in total, and looks like the following.

The `push` instruction pushes a 32bit value on the stack, and the `retn` instruction pops a 32bit address off the stack into the Instruction Pointer (in other words, it starts execution at the address which is found at the top of the stack.)

example 1

Let’s look at an example. In this case I can hook a function `WinExec` from `kernel32.dll` (`hooking.cpp`):

```
/*
hooking.cpp
basic hooking example with push/retn method
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/03/08/basic-hooking-2.html
*/
#include <windows.h>

// buffer for saving original bytes
char originalBytes[6];

FARPROC hookedAddress;

// we will jump to after the hook has been installed
int __stdcall myFunc(LPCSTR lpCmdLine, UINT uCmdShow) {
  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, originalBytes, 6,
NULL);
  return WinExec("mspaint", uCmdShow);
}

// hooking logic
void setMySuperHook() {
  HINSTANCE hLib;
  VOID *myFuncAddress;
  DWORD *rOffset;
  DWORD *hookAddress;
  DWORD src;
  DWORD dst;
  CHAR patch[6]= {0};

  // get memory address of function WinExec
  hLib = LoadLibraryA("kernel32.dll");
  hookedAddress = GetProcAddress(hLib, "WinExec");

  // save the first 6 bytes into originalBytes (buffer)
  ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 6,
NULL);

  // overwrite the first 6 bytes with a jump to myFunc
  myFuncAddress = &myFunc;

  // create a patch "push <addr>, retn"
  memcpy_s(patch, 1, "\x68", 1); // 0x68 opcode for push
  memcpy_s(patch + 1, 4, &myFuncAddress, 4);
  memcpy_s(patch + 5, 1, "\xC3", 1); // opcode for retn

  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, patch, 6, NULL);
}

int main() {

  // call original
```

```
  WinExec("notepad", SW_SHOWDEFAULT);

  // install hook
  setMySuperHook();

  // call after install hook
  WinExec("notepad", SW_SHOWDEFAULT);

}
```

As you can see, the source code is identical to the example from the first post about hooking. The only difference is:

```
37    // overwrite the first 6 bytes with a jump to myFunc
38    myFuncAddress = &myFunc;
39
40    // create a patch "push <addr>, retn"
41    memcpy_s(patch, 1, "\x68", 1); // 0x68 opcode for push
42    memcpy_s(patch + 1, 4, &myFuncAddress, 4);
43    memcpy_s(patch + 5, 1, "\xC3", 1); // opcode for retn
44
45    WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress,
46  }
```

That will translate into the following assembly instructions:

```
// push myFunc memory address onto the stack
push myFunc

// jump to myFunc
retn
```

Let's go to compile it:

```
i686-w64-mingw32-g++ -O2 hooking.cpp -o hooking.exe -mconsole -I/usr/share/mingw-
w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-
exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
>/dev/null 2>&1
```

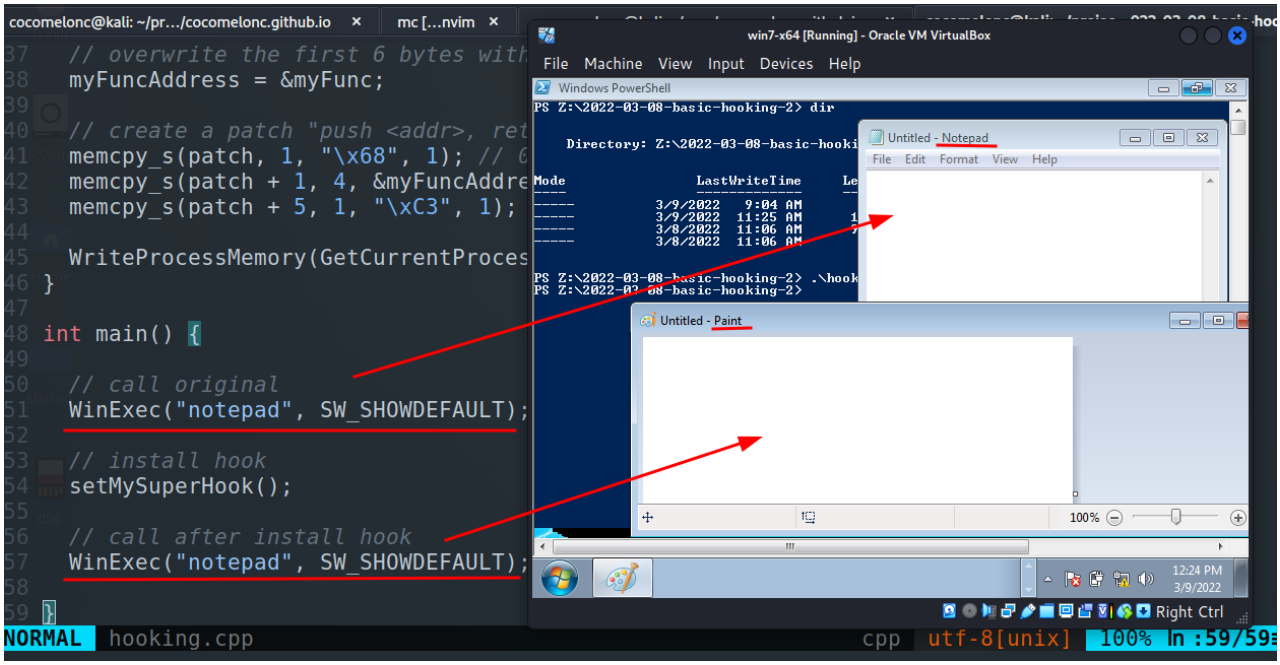
```
┌──(cocomelonc㉿kali)-[~/projects/hacking/cybersec_blog/2022-03-08-basic-hooking-2]
└─$ i686-w64-mingw32-g++ -O2 hooking.cpp -o hooking.exe -mconsole -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive >/dev/null 2>&1

┌──(cocomelonc㉿kali)-[~/projects/hacking/cybersec_blog/2022-03-08-basic-hooking-2]
└─$ ls -lht
total 116K
-rwxr-xr-x 1 cocomelonc cocomelonc  14K Mar  9 11:40 hooking.exe
-rw-r--r-- 1 cocomelonc cocomelonc 1.5K Mar  9 11:31 hooking.cpp
-rwxr-xr-x 1 cocomelonc cocomelonc  91K Mar  8 11:06 pet.dll
-rw-r--r-- 1 cocomelonc cocomelonc  900 Mar  8 11:06 pet.cpp

┌──(cocomelonc㉿kali)-[~/projects/hacking/cybersec_blog/2022-03-08-basic-hooking-2]
└─$
```

And run on `Windows 7 x64`:

```
.\hooking.exe
```



As you can see everything is worked perfectly :)

x86 API Hooking Demystified
WinExec
source code in github

> This is a practical case for educational purposes only.

Thanks for your time happy hacking and good bye!
PS. All drawings and screenshots are mine