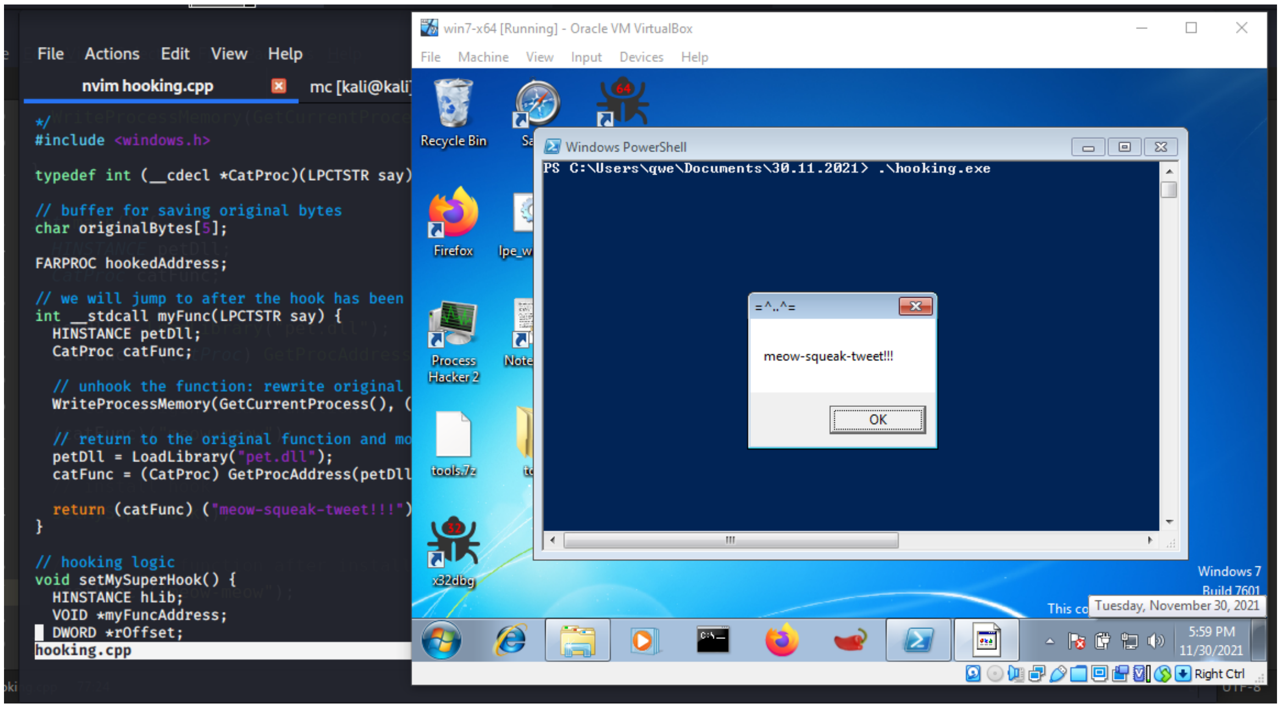
Windows API hooking. Simple C++ example.

cocomelonc.github.io/tutorial/2021/11/30/basic-hooking-1.html

November 30, 2021

5 minute read

Hello, cybersecurity enthusiasts and white hackers!



```
*/
#include <windows.h>

typedef int (__cdecl *CatProc)(LPCTSTR say)

// buffer for saving original bytes
char originalBytes[5];

FARPROC hookedAddress;

// we will jump to after the hook has been
int __stdcall myFunc(LPCTSTR say) {
  HINSTANCE petDll;
  CatProc catFunc;

  // unhook the function: rewrite original
  WriteProcessMemory(GetCurrentProcess(), (

  // return to the original function and mo
  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress(petDll

  return (catFunc) ("meow-squeak-tweet!!!")
}

// hooking logic
void setMySuperHook() {
  HINSTANCE hLib;
  VOID *myFuncAddress;
  DWORD *rOffset;
hooking.cpp
```

```
PS C:\Users\qwe\Documents\30.11.2021> .\hooking.exe
```

```
meow-squeak-tweet!!!
```

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.

example 1

Before hooking windows API functions I will consider the case of how to do this with an exported function from a DLL.

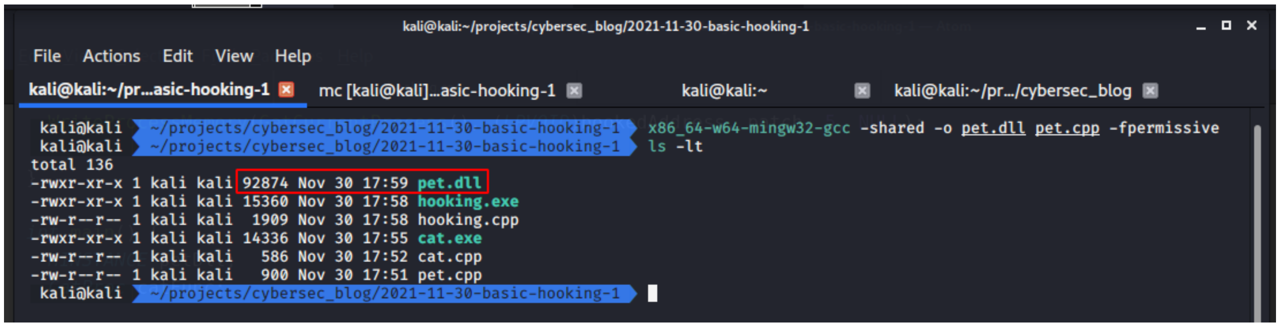
For example we have DLL with this logic (`pet.cpp`):

```
/*
pet.dll - DLL example for basic hooking
*/

#include <windows.h>
#pragma comment (lib, "user32.lib")

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

extern "C" {
  __declspec(dllexport) int _cdecl Cat(LPCTSTR say) {
    MessageBox(NULL, say, "=^..^=", MB_OK);
          return 1;
        }
}

extern "C" {
  __declspec(dllexport) int _cdecl Mouse(LPCTSTR say) {
    MessageBox(NULL, say, "<:3()~~", MB_OK);
          return 1;
        }
}

extern "C" {
  __declspec(dllexport) int _cdecl Frog(LPCTSTR say) {
    MessageBox(NULL, say, "8)~", MB_OK);
          return 1;
        }
}

extern "C" {
  __declspec(dllexport) int _cdecl Bird(LPCTSTR say) {
    MessageBox(NULL, say, "<(-)", MB_OK);
          return 1;
        }
}
```

As you can see this DLL have simplest exported functions: `Cat`, `Mouse`, `Frog`, `Bird` with one param `say`. As you can see the logic of this functions is simplest, just pop-up message with title.

Let's go to compile it:

```
x86_64-w64-mingw32-gcc -shared -o pet.dll pet.cpp -fpermissive
```


kali@kali:~/projects/cybersec_blog/2021-11-30-basic-hooking-1
File Actions Edit View Help
kali@kali:~/pr...asic-hooking-1
mc [kali@kali]...asic-hooking-1
kali@kali:~
kali@kali:~/pr.../cybersec_blog

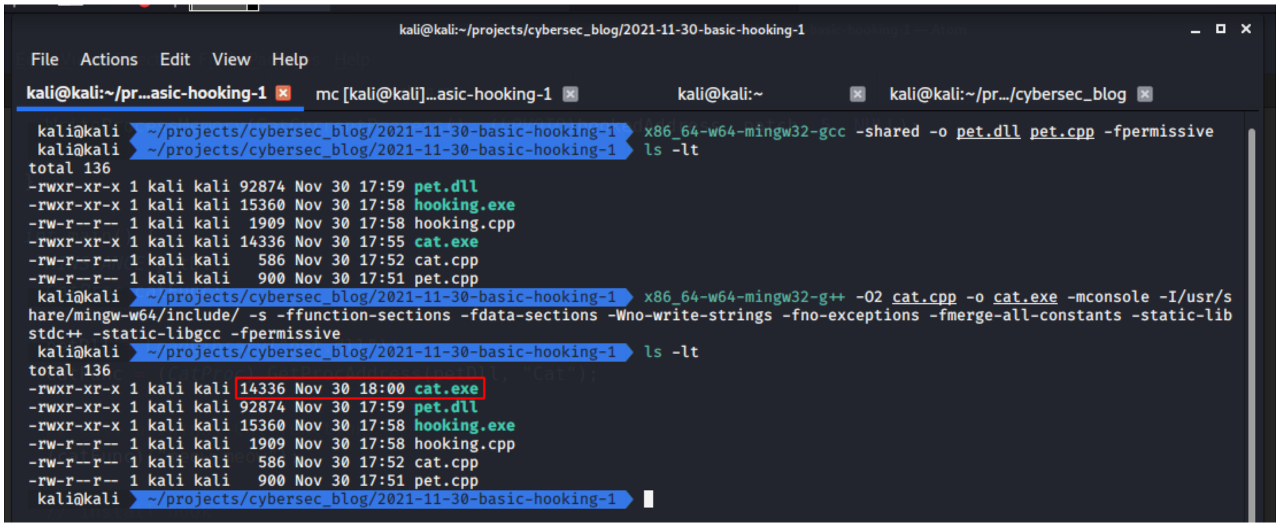
```
kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  x86_64-w64-mingw32-gcc -shared -o pet.dll pet.cpp -fpermissive
kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  ls -lt
total 136
-rwxr-xr-x 1 kali kali 92874 Nov 30 17:59 pet.dll
-rwxr-xr-x 1 kali kali 15360 Nov 30 17:58 hooking.exe
-rw-r--r-- 1 kali kali  1909 Nov 30 17:58 hooking.cpp
-rwxr-xr-x 1 kali kali 14336 Nov 30 17:55 cat.exe
-rw-r--r-- 1 kali kali   586 Nov 30 17:52 cat.cpp
-rw-r--r-- 1 kali kali   900 Nov 30 17:51 pet.cpp
kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1 
```

and then, create a simple code to validate this DLL (`cat.cpp`):

```
#include <windows.h>

typedef int (__cdecl *CatProc)(LPCTSTR say);
typedef int (__cdecl *BirdProc)(LPCTSTR say);

int main(void) {
  HINSTANCE petDll;
  CatProc catFunc;
  BirdProc birdFunc;
  BOOL freeRes;

  petDll = LoadLibrary("pet.dll");

  if (petDll != NULL) {
    catFunc = (CatProc) GetProcAddress(petDll, "Cat");
    birdFunc = (BirdProc) GetProcAddress(petDll, "Bird");
    if ((catFunc != NULL) && (birdFunc != NULL)) {
      (catFunc) ("meow-meow");
      (catFunc) ("mmmmeow");
      (birdFunc) ("tweet-tweet");
    }
    freeRes = FreeLibrary(petDll);
  }

  return 0;
}
```

Let's go to compile it:

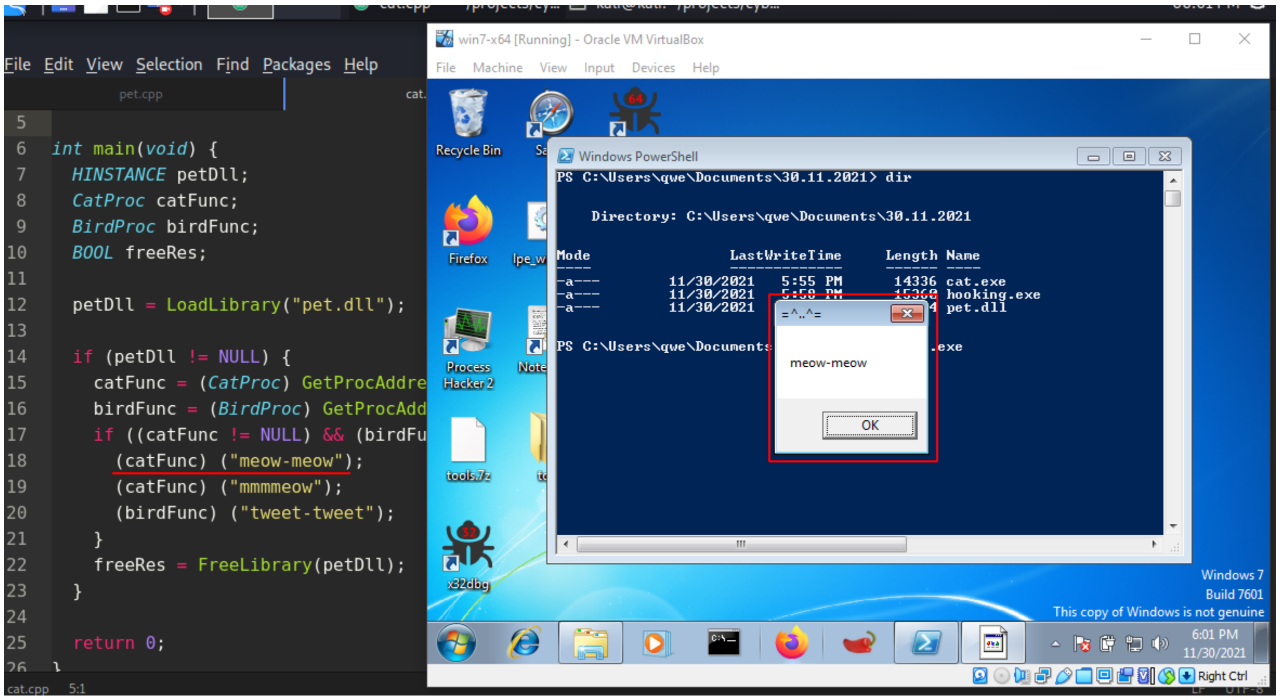
```
x86_64-w64-mingw32-g++ -O2 cat.cpp -o cat.exe -mconsole -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```


kali@kali:~/projects/cybersec_blog/2021-11-30-basic-hooking-1
File Actions Edit View Help
kali@kali:~/pr...asic-hooking-1
mc [kali@kali]...asic-hooking-1
kali@kali:~
kali@kali:~/pr.../cybersec_blog

```
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  x86_64-w64-mingw32-gcc -shared -o pet.dll pet.cpp -fpermissive
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  ls -lt
total 136
-rwxr-xr-x 1 kali kali 92874 Nov 30 17:59 pet.dll
-rwxr-xr-x 1 kali kali 15360 Nov 30 17:58 hooking.exe
-rw-r--r-- 1 kali kali  1909 Nov 30 17:58 hooking.cpp
-rwxr-xr-x 1 kali kali 14336 Nov 30 17:55 cat.exe
-rw-r--r-- 1 kali kali   586 Nov 30 17:52 cat.cpp
-rw-r--r-- 1 kali kali   900 Nov 30 17:51 pet.cpp
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  x86_64-w64-mingw32-g++ -O2 cat.cpp -o cat.exe -mconsole -I/usr/s
hare/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-lib
stdc++ -static-libgcc -fpermissive
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  ls -lt
total 136
-rwxr-xr-x 1 kali kali 14336 Nov 30 18:00 cat.exe
-rwxr-xr-x 1 kali kali 92874 Nov 30 17:59 pet.dll
-rwxr-xr-x 1 kali kali 15360 Nov 30 17:58 hooking.exe
-rw-r--r-- 1 kali kali  1909 Nov 30 17:58 hooking.cpp
-rw-r--r-- 1 kali kali   586 Nov 30 17:52 cat.cpp
-rw-r--r-- 1 kali kali   900 Nov 30 17:51 pet.cpp
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1 
```

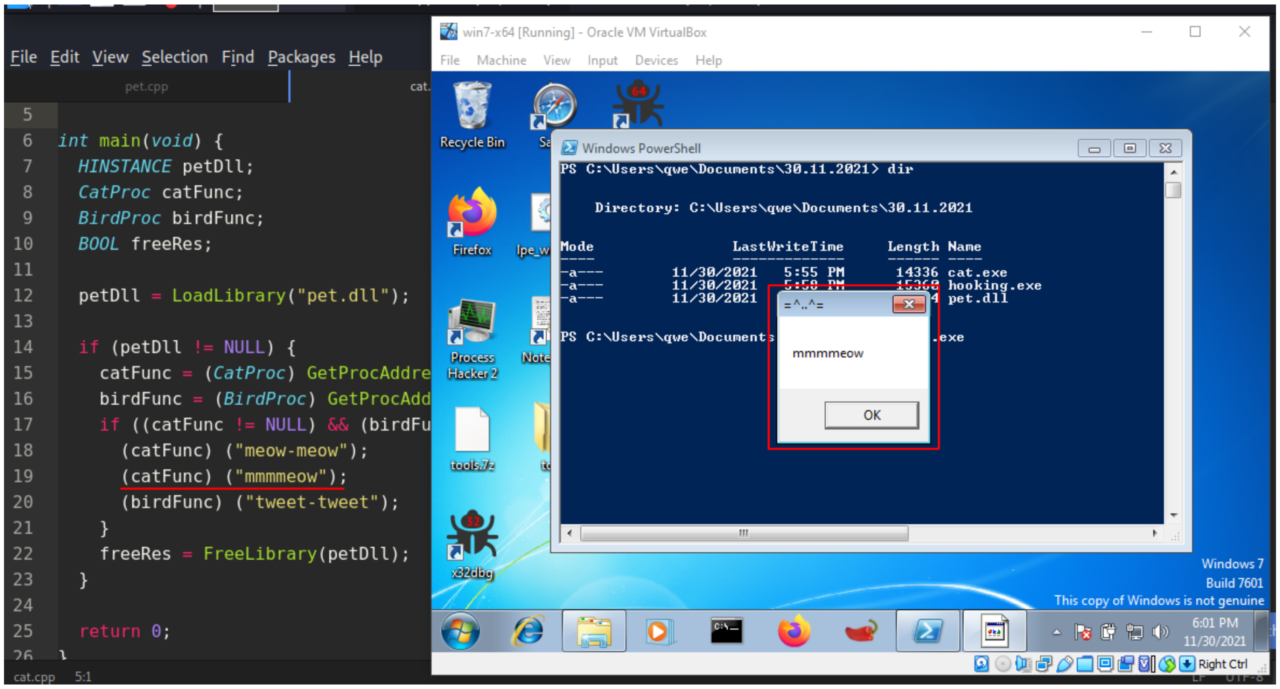
and run on `Windows 7 x64`:

```
.\cat.exe
```


File Edit View Selection Find Packages Help
pet.cpp
cat.
win7-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Recycle Bin
Firefox
Process Hacker 2
tools.7z
x32dbg
Windows PowerShell
=^..^=
meow-meow
OK
Windows 7
Build 7601
This copy of Windows is not genuine
6:01 PM
11/30/2021
Right Ctrl
cat.cpp 5:1

```
int main(void) {
  HINSTANCE petDll;
  CatProc catFunc;
  BirdProc birdFunc;
  BOOL freeRes;

  petDll = LoadLibrary("pet.dll");

  if (petDll != NULL) {
    catFunc = (CatProc) GetProcAddre
    birdFunc = (BirdProc) GetProcAdd
    if ((catFunc != NULL) && (birdFu
      (catFunc) ("meow-meow");
      (catFunc) ("mmmmeow");
      (birdFunc) ("tweet-tweet");
    }
    freeRes = FreeLibrary(petDll);
  }

  return 0;
}
```

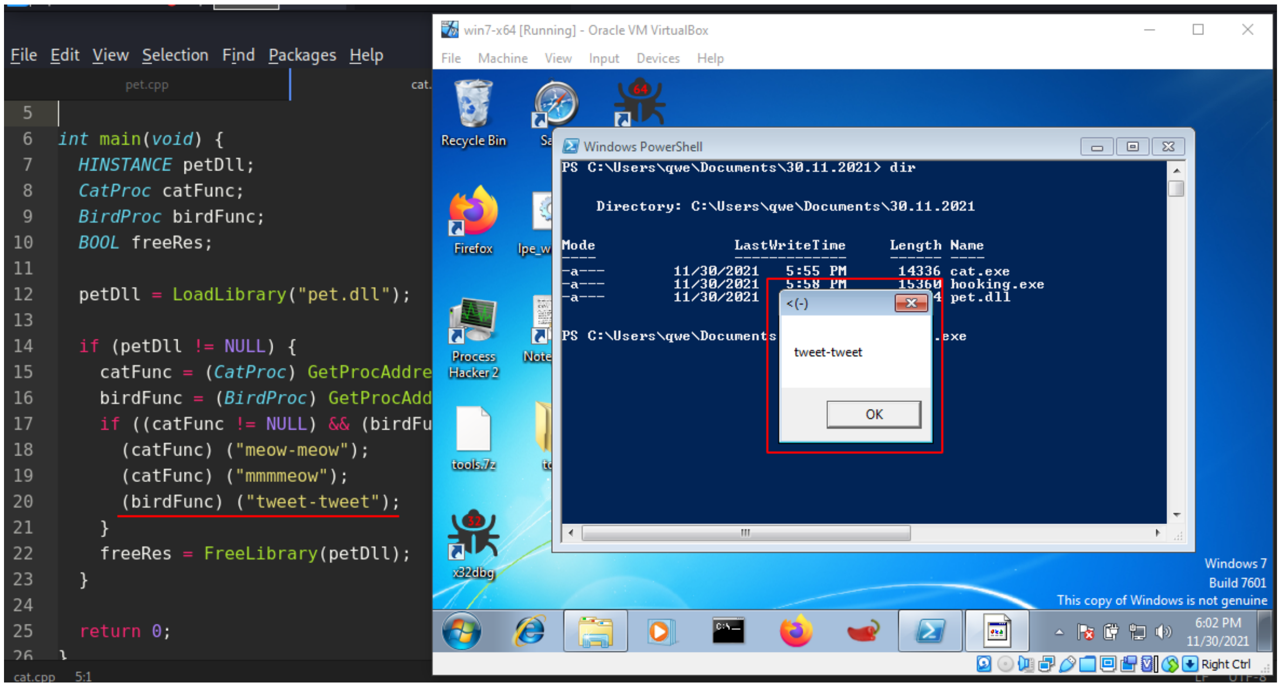
```
PS C:\Users\qwe\Documents\30.11.2021> dir


    Directory: C:\Users\qwe\Documents\30.11.2021


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        11/30/2021   5:55 PM      14336 cat.exe
-a---        11/30/2021   5:58 PM      15360 hooking.exe
-a---        11/30/2021                    4 pet.dll


PS C:\Users\qwe\Documents                    .exe
```



```
int main(void) {
  HINSTANCE petDll;
  CatProc catFunc;
  BirdProc birdFunc;
  BOOL freeRes;

  petDll = LoadLibrary("pet.dll");

  if (petDll != NULL) {
    catFunc = (CatProc) GetProcAddre
    birdFunc = (BirdProc) GetProcAdd
    if ((catFunc != NULL) && (birdFu
      (catFunc) ("meow-meow");
      (catFunc) ("mmmmeow");
      (birdFunc) ("tweet-tweet");
    }
    freeRes = FreeLibrary(petDll);
  }

  return 0;
}
```

```
PS C:\Users\qwe\Documents\30.11.2021> dir

    Directory: C:\Users\qwe\Documents\30.11.2021

Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        11/30/2021   5:55 PM      14336 cat.exe
-a---        11/30/2021   5:58 PM      15360 hooking.exe
-a---        11/30/2021                    4 pet.dll

PS C:\Users\qwe\Document                        .exe
```

```
=^..^=
mmmmeow
OK
```



```
int main(void) {
  HINSTANCE petDll;
  CatProc catFunc;
  BirdProc birdFunc;
  BOOL freeRes;

  petDll = LoadLibrary("pet.dll");

  if (petDll != NULL) {
    catFunc = (CatProc) GetProcAddre
    birdFunc = (BirdProc) GetProcAdd
    if ((catFunc != NULL) && (birdFu
      (catFunc) ("meow-meow");
      (catFunc) ("mmmmeow");
      (birdFunc) ("tweet-tweet");
    }
    freeRes = FreeLibrary(petDll);
  }

  return 0;
}
```

```
PS C:\Users\qwe\Documents\30.11.2021> dir

    Directory: C:\Users\qwe\Documents\30.11.2021

Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        11/30/2021   5:55 PM      14336 cat.exe
-a---        11/30/2021   5:58 PM      15360 hooking.exe
-a---        11/30/2021                    4 pet.dll

PS C:\Users\qwe\Document                        .exe
```

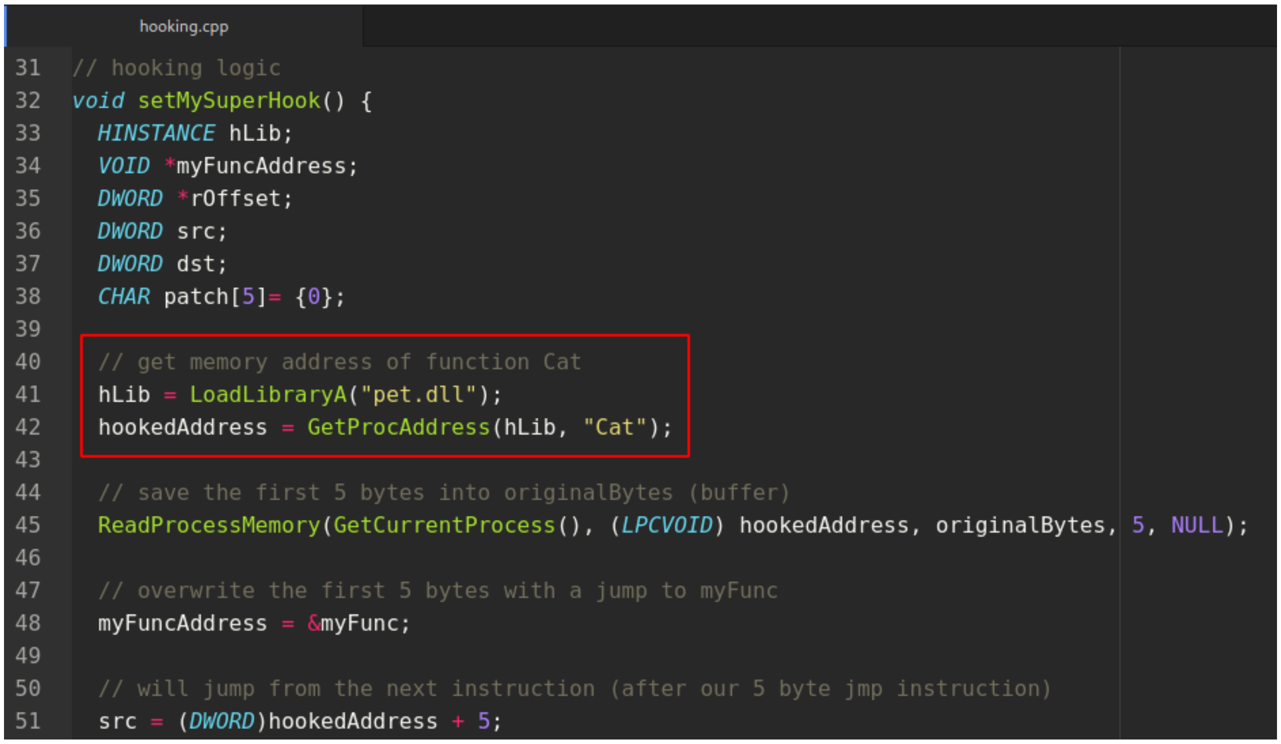
```
<(-)
tweet-tweet
OK
```

and as you can see, everything works as expected.

Then, for example `Cat` function will be hooked in this scenario, but it could be any.

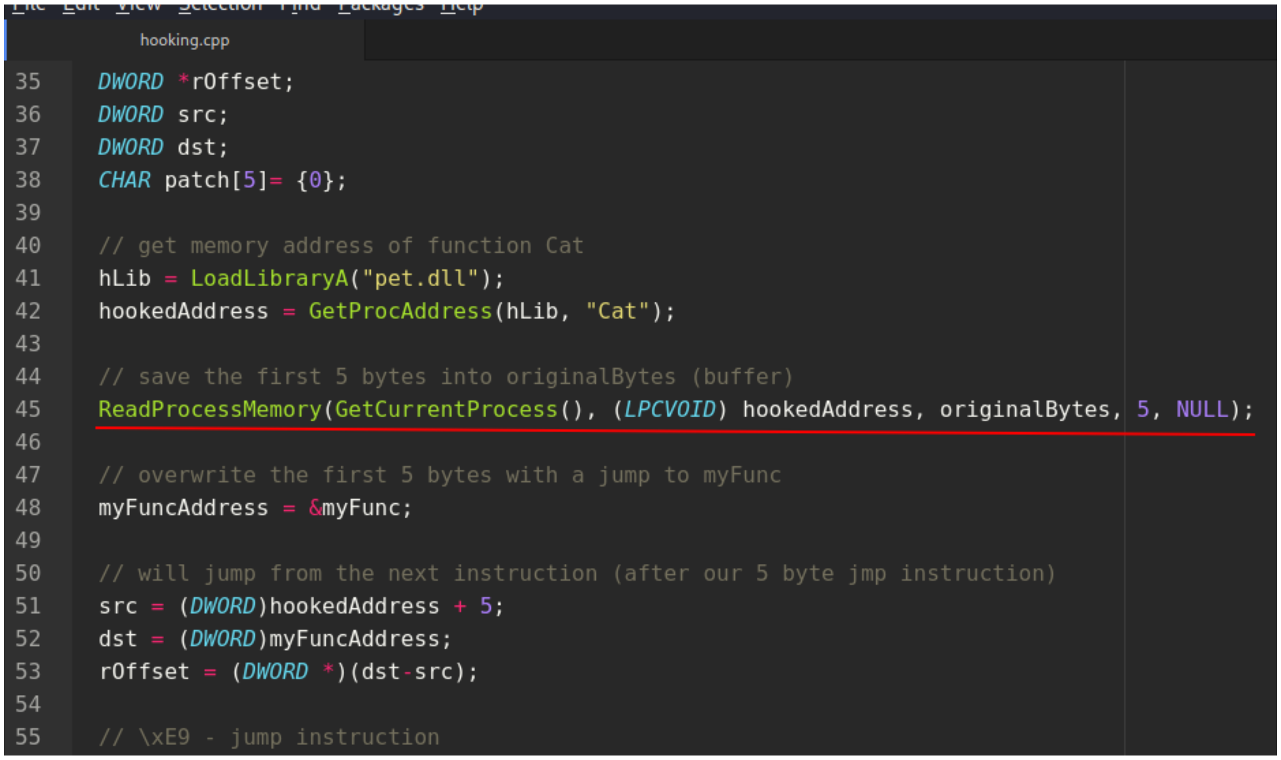
The workflow of this technique is as follows:

First, get memory address of the `Cat` function.



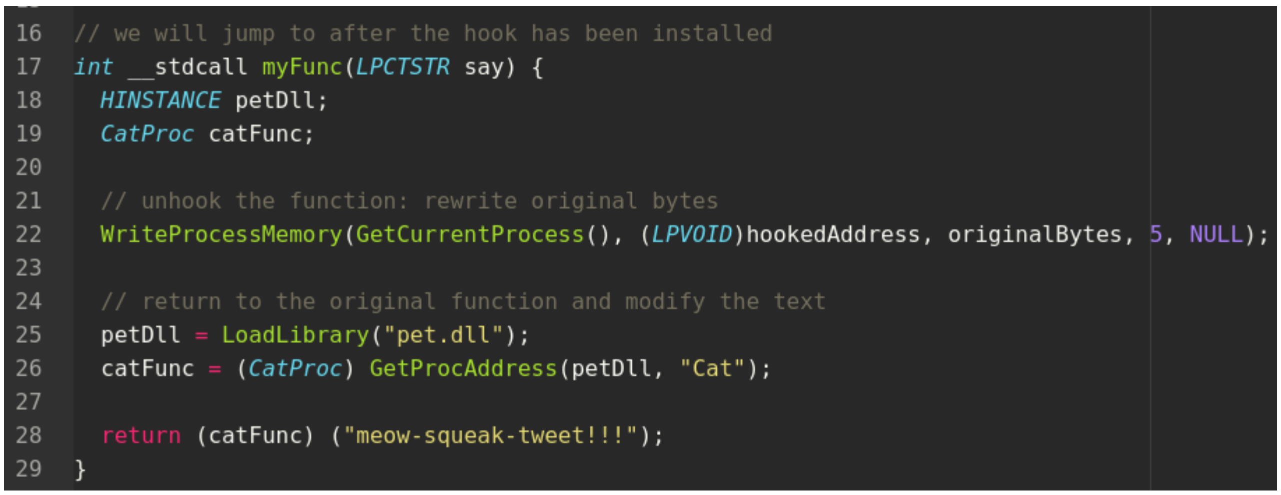
```cpp
// hooking logic
void setMySuperHook() {
  HINSTANCE hLib;
  VOID *myFuncAddress;
  DWORD *rOffset;
  DWORD src;
  DWORD dst;
  CHAR patch[5]= {0};

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

  // save the first 5 bytes into originalBytes (buffer)
  ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5, NULL);

  // overwrite the first 5 bytes with a jump to myFunc
  myFuncAddress = &myFunc;

  // will jump from the next instruction (after our 5 byte jmp instruction)
  src = (DWORD)hookedAddress + 5;
```

then, save the first 5 bytes of the `Cat` function. We will need this bytes:



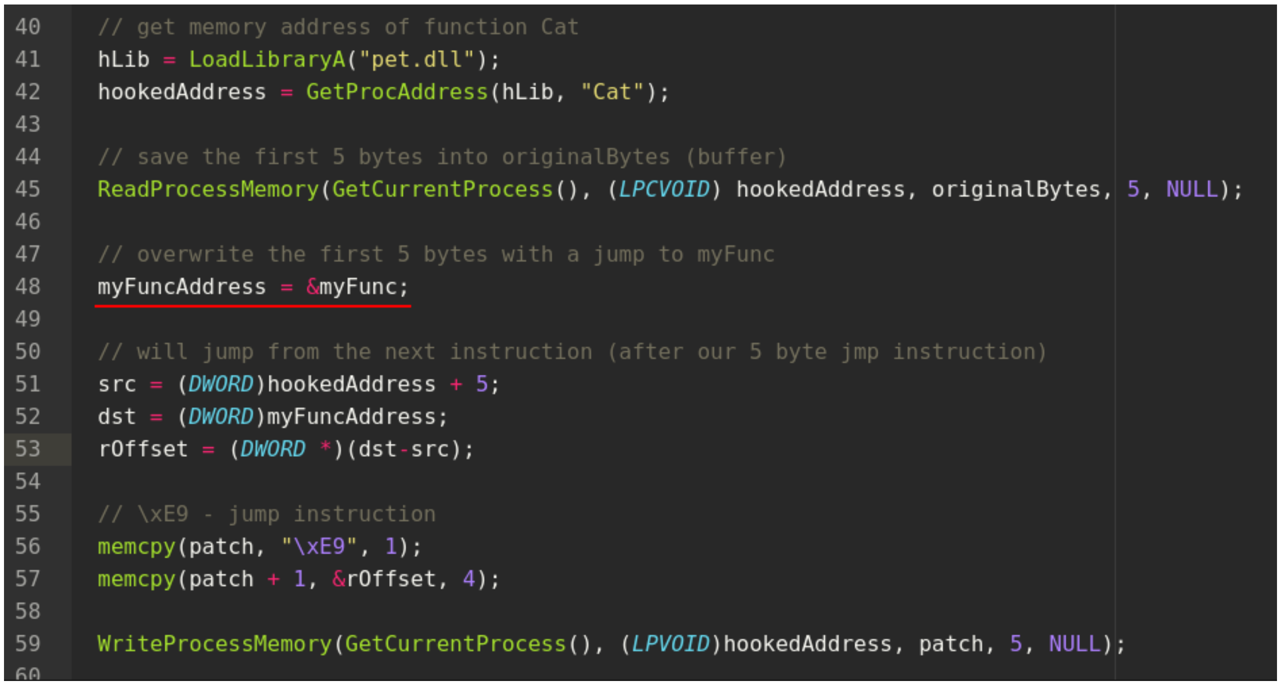
```cpp
  DWORD *rOffset;
  DWORD src;
  DWORD dst;
  CHAR patch[5]= {0};

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

  // save the first 5 bytes into originalBytes (buffer)
  ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5, NULL);

  // overwrite the first 5 bytes with a jump to myFunc
  myFuncAddress = &myFunc;

  // will jump from the next instruction (after our 5 byte jmp instruction)
  src = (DWORD)hookedAddress + 5;
  dst = (DWORD)myFuncAddress;
  rOffset = (DWORD *)(dst-src);

  // \xE9 - jump instruction
```

then, create a `myFunc` function that will be executed when the original `Cat` is called:



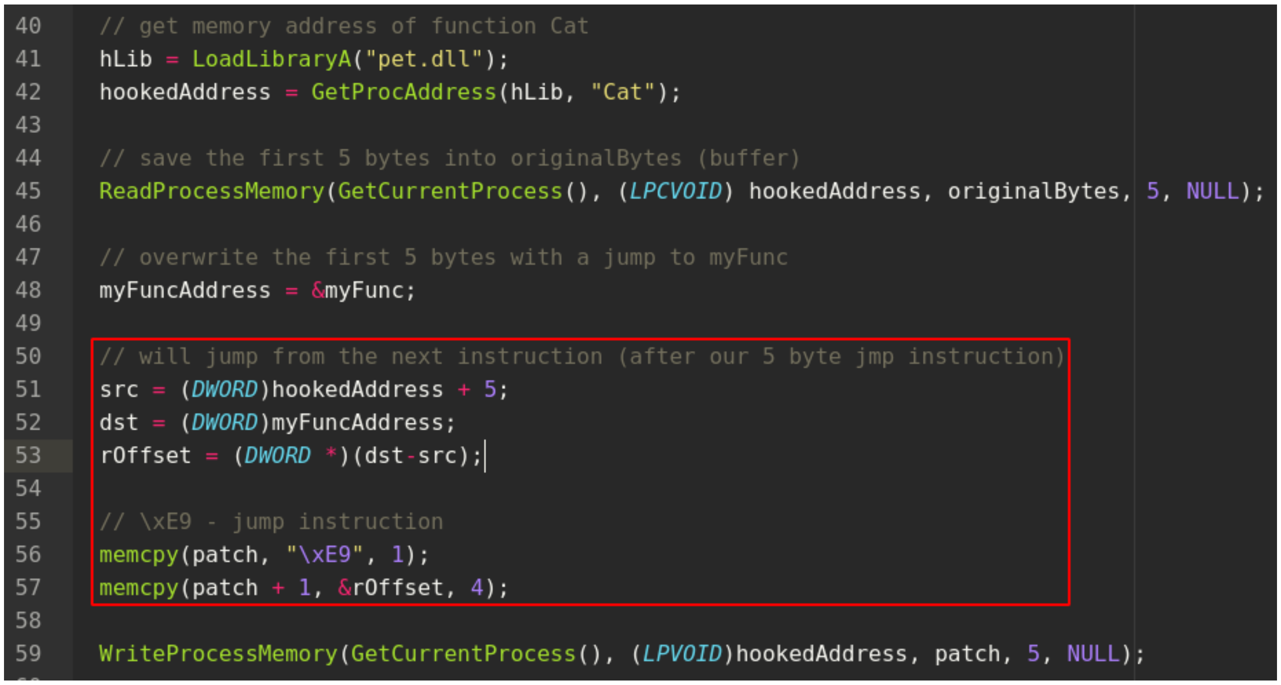
```
// we will jump to after the hook has been installed
int __stdcall myFunc(LPCTSTR say) {
  HINSTANCE petDll;
  CatProc catFunc;

  // unhook the function: rewrite original bytes
  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, originalBytes, 5, NULL);

  // return to the original function and modify the text
  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress(petDll, "Cat");

  return (catFunc) ("meow-squeak-tweet!!!");
}
```

overwrite `5` bytes with a jump to `myFunc`:



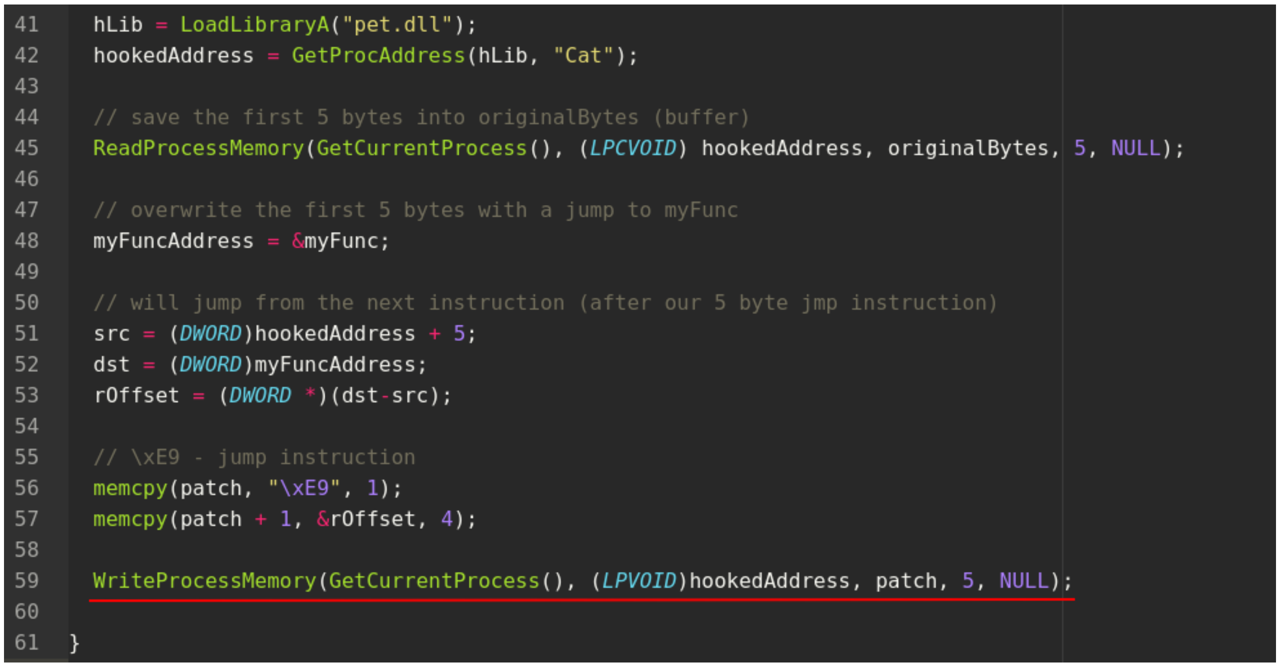
```
  // get memory address of function Cat
  hLib = LoadLibraryA("pet.dll");
  hookedAddress = GetProcAddress(hLib, "Cat");

  // save the first 5 bytes into originalBytes (buffer)
  ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5, NULL);

  // overwrite the first 5 bytes with a jump to myFunc
  myFuncAddress = &myFunc;

  // will jump from the next instruction (after our 5 byte jmp instruction)
  src = (DWORD)hookedAddress + 5;
  dst = (DWORD)myFuncAddress;
  rOffset = (DWORD *)(dst-src);

  // \xE9 - jump instruction
  memcpy(patch, "\xE9", 1);
  memcpy(patch + 1, &rOffset, 4);

  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, patch, 5, NULL);
```

Then, create a “patch”:



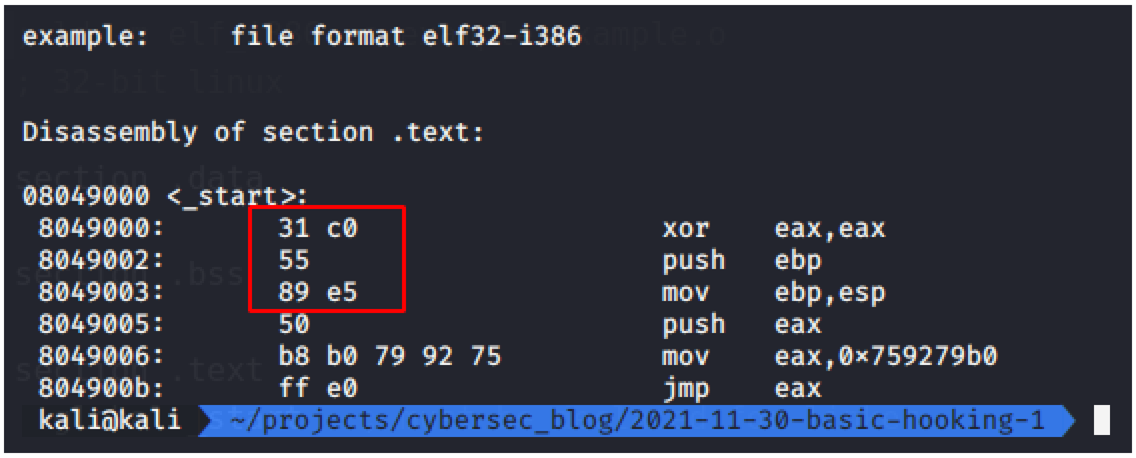
```
// get memory address of function Cat
hLib = LoadLibraryA("pet.dll");
hookedAddress = GetProcAddress(hLib, "Cat");

// save the first 5 bytes into originalBytes (buffer)
ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5, NULL);

// overwrite the first 5 bytes with a jump to myFunc
myFuncAddress = &myFunc;

// will jump from the next instruction (after our 5 byte jmp instruction)
src = (DWORD)hookedAddress + 5;
dst = (DWORD)myFuncAddress;
rOffset = (DWORD *)(dst-src);

// \xE9 - jump instruction
memcpy(patch, "\xE9", 1);
memcpy(patch + 1, &rOffset, 4);

WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, patch, 5, NULL);
```

in the next step, patch our `Cat` function (redirect `Cat` function to `myFunc`):



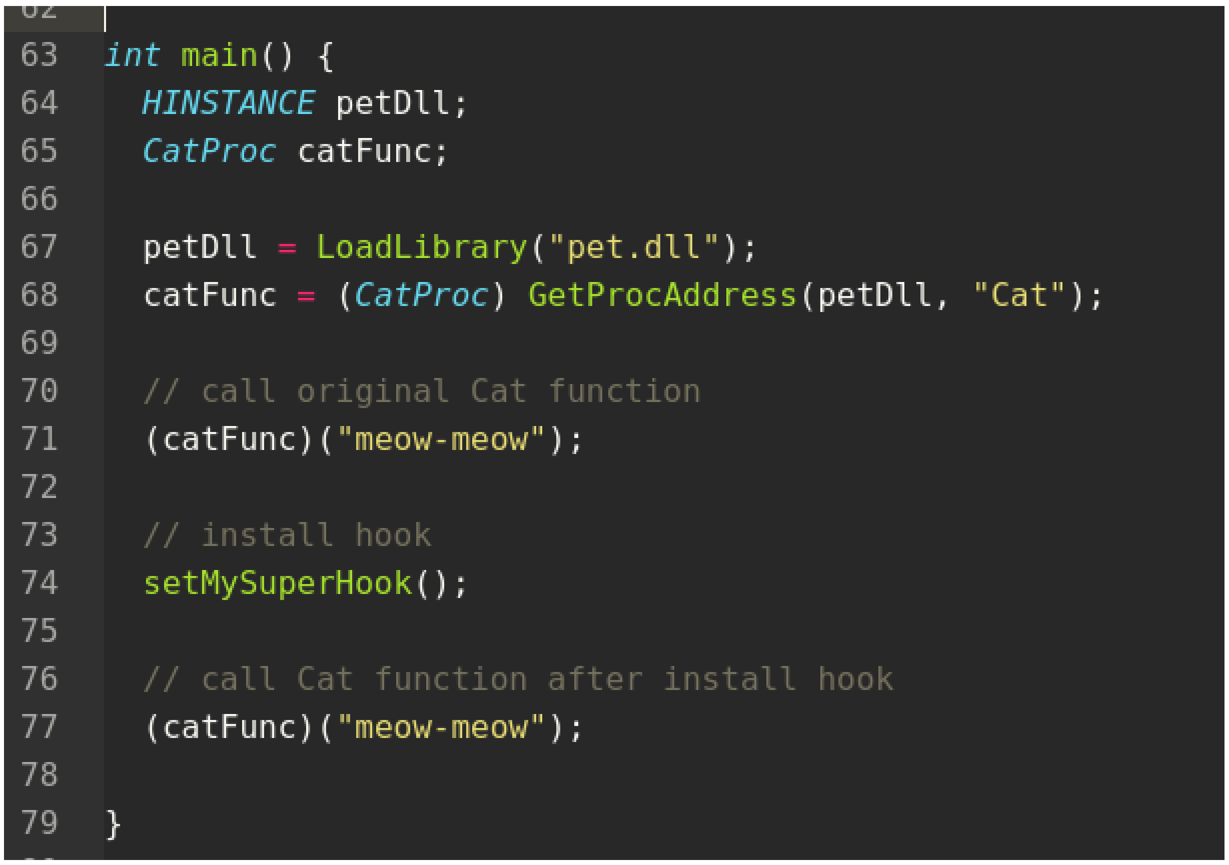
```
hLib = LoadLibraryA("pet.dll");
hookedAddress = GetProcAddress(hLib, "Cat");

// save the first 5 bytes into originalBytes (buffer)
ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5, NULL);

// overwrite the first 5 bytes with a jump to myFunc
myFuncAddress = &myFunc;

// will jump from the next instruction (after our 5 byte jmp instruction)
src = (DWORD)hookedAddress + 5;
dst = (DWORD)myFuncAddress;
rOffset = (DWORD *)(dst-src);

// \xE9 - jump instruction
memcpy(patch, "\xE9", 1);
memcpy(patch + 1, &rOffset, 4);

WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, patch, 5, NULL);

}
```

So what have we done here? This trick is *"classic 5-byte hook"* technique. If we disassemble function:


```
example:     file format elf32-i386

Disassembly of section .text:

08049000 <_start>:
 8049000:       31 c0                   xor    eax,eax
 8049002:       55                      push   ebp
 8049003:       89 e5                   mov    ebp,esp
 8049005:       50                      push   eax
 8049006:       b8 b0 79 92 75          mov    eax,0x759279b0
 804900b:       ff e0                   jmp    eax
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1
```

The highlighted `5` bytes is a fairly typical prologue found in many API functions. By overwriting these first `5` bytes with a `jmp` instruction, we are redirecting execution to our own defined function. We will save the original bytes so that they can be referenced later when we want to pass execution back to the hooked function.

So firstly, we call original `Cat` function, set our hook and call `Cat` again:

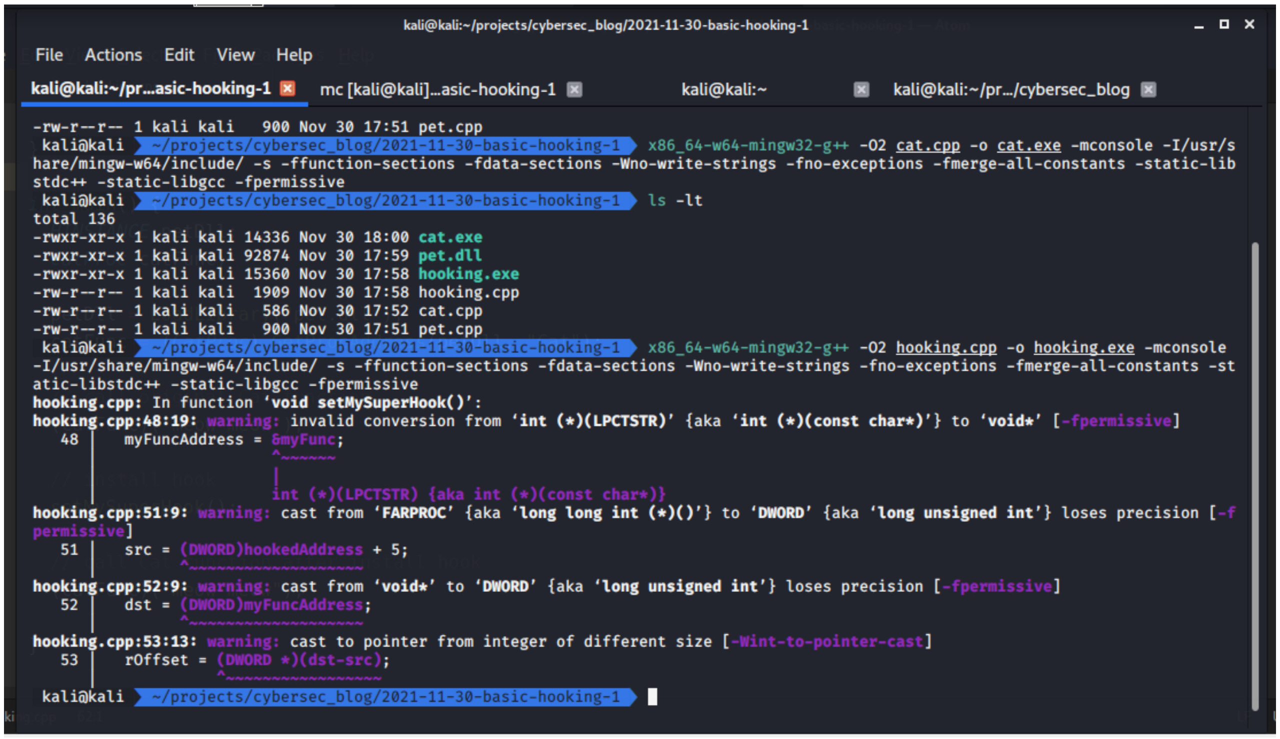

```
int main() {
  HINSTANCE petDll;
  CatProc catFunc;

  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress(petDll, "Cat");

  // call original Cat function
  (catFunc)("meow-meow");

  // install hook
  setMySuperHook();

  // call Cat function after install hook
  (catFunc)("meow-meow");

}
```

Full source code is:

```
/*
hooking.cpp
basic hooking example
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/11/30/basic-hooking-1.html
*/
#include <windows.h>

typedef int (__cdecl *CatProc)(LPCTSTR say);

// buffer for saving original bytes
char originalBytes[5];

FARPROC hookedAddress;

// we will jump to after the hook has been installed
int __stdcall myFunc(LPCTSTR say) {
  HINSTANCE petDll;
  CatProc catFunc;

  // unhook the function: rewrite original bytes
  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, originalBytes, 5,
NULL);

  // return to the original function and modify the text
  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress(petDll, "Cat");

  return (catFunc) ("meow-squeak-tweet!!!");
}

// hooking logic
void setMySuperHook() {
  HINSTANCE hLib;
  VOID *myFuncAddress;
  DWORD *rOffset;
  DWORD src;
  DWORD dst;
  CHAR patch[5]= {0};

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

  // save the first 5 bytes into originalBytes (buffer)
  ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5,
NULL);

  // overwrite the first 5 bytes with a jump to myFunc
  myFuncAddress = &myFunc;

  // will jump from the next instruction (after our 5 byte jmp instruction)
  src = (DWORD)hookedAddress + 5;
  dst = (DWORD)myFuncAddress;
  rOffset = (DWORD *)(dst-src);

  // \xE9 - jump instruction
  memcpy(patch, "\xE9", 1);
  memcpy(patch + 1, &rOffset, 4);

  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, patch, 5, NULL);

}

int main() {
  HINSTANCE petDll;
  CatProc catFunc;

  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress(petDll, "Cat");

  // call original Cat function
  (catFunc)("meow-meow");

  // install hook
  setMySuperHook();

  // call Cat function after install hook
  (catFunc)("meow-meow");

}
```

Let’s go to compile this:

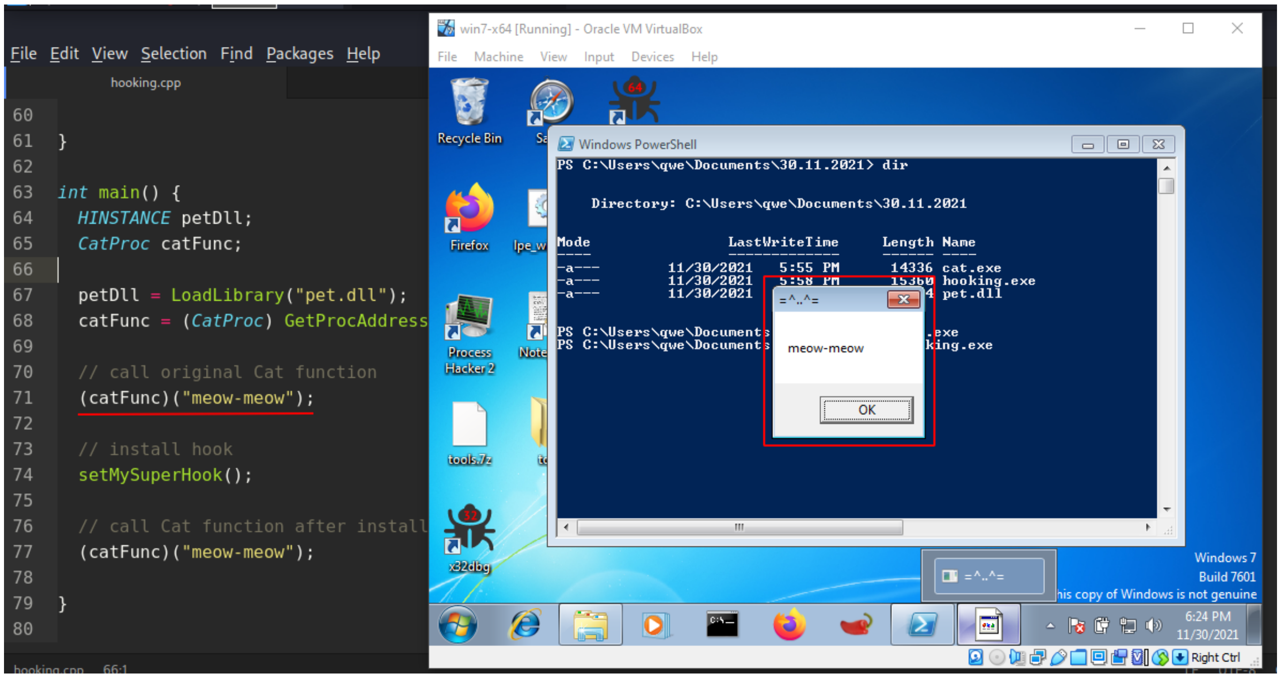
```
x86_64-w64-mingw32-g++ -O2 hooking.cpp -o hooking.exe -mconsole -I/usr/share/mingw-
w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-
exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```


kali@kali:~/projects/cybersec_blog/2021-11-30-basic-hooking-1
File Actions Edit View Help
kali@kali:~/pr...asic-hooking-1
mc [kali@kali]...asic-hooking-1
kali@kali:~
kali@kali:~/pr.../cybersec_blog

```
-rw-r--r-- 1 kali kali   900 Nov 30 17:51 pet.cpp
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  x86_64-w64-mingw32-g++ -O2 cat.cpp -o cat.exe -mconsole -I/usr/s
hare/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-lib
stdc++ -static-libgcc -fpermissive
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  ls -lt
total 136
-rwxr-xr-x 1 kali kali 14336 Nov 30 18:00 cat.exe
-rwxr-xr-x 1 kali kali 92874 Nov 30 17:59 pet.dll
-rwxr-xr-x 1 kali kali 15360 Nov 30 17:58 hooking.exe
-rw-r--r-- 1 kali kali  1909 Nov 30 17:58 hooking.cpp
-rw-r--r-- 1 kali kali   586 Nov 30 17:52 cat.cpp
-rw-r--r-- 1 kali kali   900 Nov 30 17:51 pet.cpp
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  x86_64-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 -st
atic-libstdc++ -static-libgcc -fpermissive
hooking.cpp: In function ‘void setMySuperHook()’:
hooking.cpp:48:19: warning: invalid conversion from ‘int (*)(LPCTSTR)’ {aka ‘int (*)(const char*)’} to ‘void*’ [-fpermissive]
   48 |     myFuncAddress = &myFunc;
      |                     ^~~~~~~
      |                     |
      |                     int (*)(LPCTSTR) {aka int (*)(const char*)}
hooking.cpp:51:9: warning: cast from ‘FARPROC’ {aka ‘long long int (*)()’} to ‘DWORD’ {aka ‘long unsigned int’} loses precision [-f
permissive]
   51 |     src = (DWORD)hookedAddress + 5;
      |           ^~~~~~~~~~~~~~~~~~~~
hooking.cpp:52:9: warning: cast from ‘void*’ to ‘DWORD’ {aka ‘long unsigned int’} loses precision [-fpermissive]
   52 |     dst = (DWORD)myFuncAddress;
      |           ^~~~~~~~~~~~~~~~~~~~
hooking.cpp:53:13: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   53 |     rOffset = (DWORD *)(dst-src);
      |               ^~~~~~~~~~~~~~~~~~
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1 
```

And see it in action (on `Windows 7 x64` in this case):

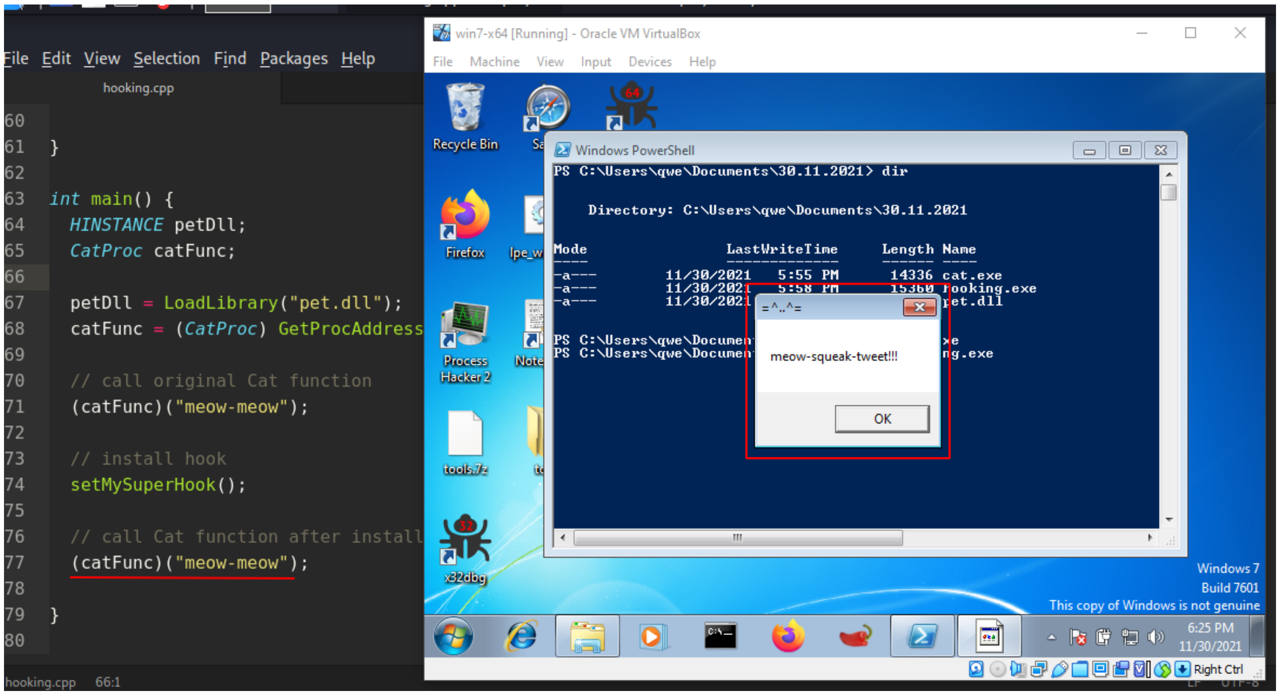
```
.\hooking.exe
```


File Edit View Selection Find Packages Help
hooking.cpp
win7-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Recycle Bin
Firefox
Process Hacker 2
tools.7z
x32dbg
Windows PowerShell
=^..^=
OK
Windows 7
Build 7601
This copy of Windows is not genuine
6:24 PM
11/30/2021
Right Ctrl

```
}

int main() {
  HINSTANCE petDll;
  CatProc catFunc;

  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress

  // call original Cat function
  (catFunc)("meow-meow");

  // install hook
  setMySuperHook();

  // call Cat function after install
  (catFunc)("meow-meow");

}
```

```
PS C:\Users\qwe\Documents\30.11.2021> dir


    Directory: C:\Users\qwe\Documents\30.11.2021


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        11/30/2021   5:55 PM      14336 cat.exe
-a---        11/30/2021   5:58 PM      15360 hooking.exe
-a---        11/30/2021                    4 pet.dll


PS C:\Users\qwe\Documents                   .exe
PS C:\Users\qwe\Documents                   king.exe
```

meow-meow



```
}

int main() {
  HINSTANCE petDll;
  CatProc catFunc;

  petDll = LoadLibrary("pet.dll");
  catFunc = (CatProc) GetProcAddress

  // call original Cat function
  (catFunc)("meow-meow");

  // install hook
  setMySuperHook();

  // call Cat function after install
  (catFunc)("meow-meow");

}
```

```
PS C:\Users\qwe\Documents\30.11.2021> dir


    Directory: C:\Users\qwe\Documents\30.11.2021


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        11/30/2021   5:55 PM      14336 cat.exe
-a---        11/30/2021   5:58 PM      15360 hooking.exe
-a---        11/30/2021                      pet.dll


PS C:\Users\qwe\Documen                      xe
PS C:\Users\qwe\Documen                      ng.exe
```

```
=^..^=
meow-squeak-tweet!!!
OK
```

As you can see our hook is worked perfectly!! Cat goes `meow-squeak-tweet!!!` instead `meow-meow`!

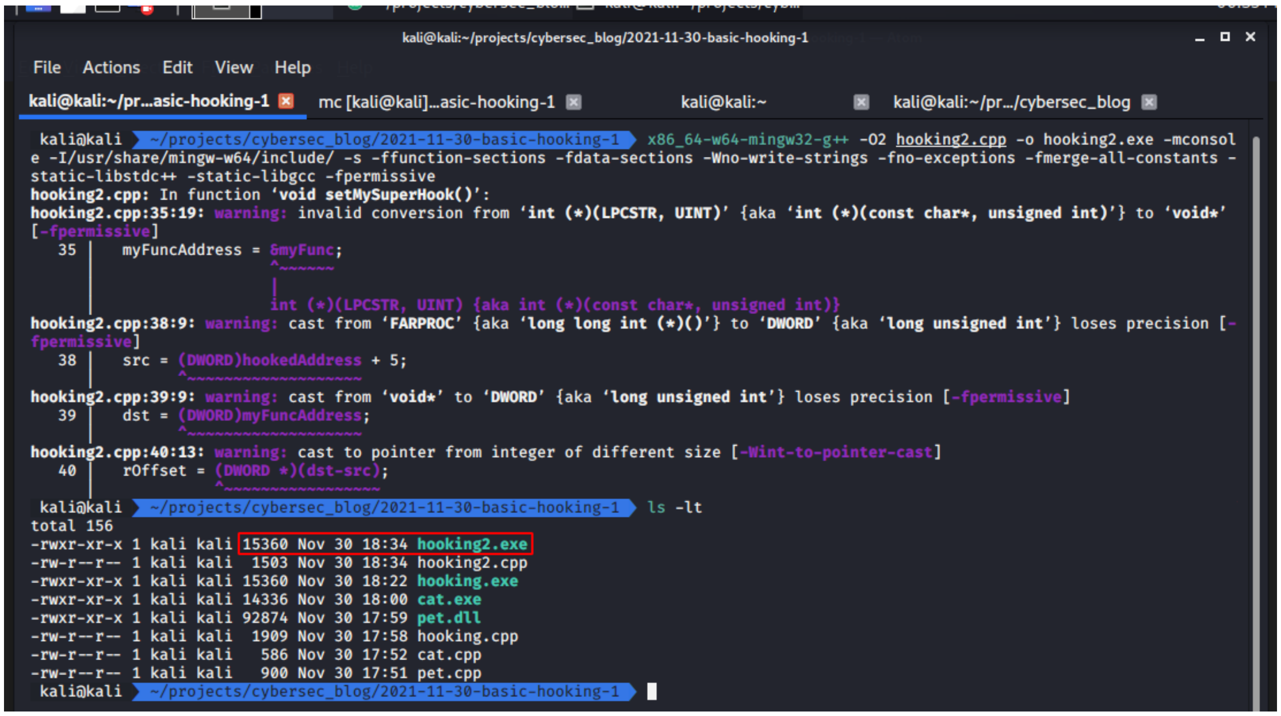
example 2

Similarly, you can hook for example, a function `WinExec` from `kernel32.dll` (`hooking2.cpp`):

```
#include <windows.h>

// buffer for saving original bytes
char originalBytes[5];

FARPROC hookedAddress;

// we will jump to after the hook has been installed
int __stdcall myFunc(LPCSTR lpCmdLine, UINT uCmdShow) {

  // unhook the function: rewrite original bytes
  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, originalBytes, 5,
NULL);

  // return to the original function and modify the text
  return WinExec("calc", uCmdShow);
}

// hooking logic
void setMySuperHook() {
  HINSTANCE hLib;
  VOID *myFuncAddress;
  DWORD *rOffset;
  DWORD src;
  DWORD dst;
  CHAR patch[5]= {0};

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

  // save the first 5 bytes into originalBytes (buffer)
  ReadProcessMemory(GetCurrentProcess(), (LPCVOID) hookedAddress, originalBytes, 5,
NULL);

  // overwrite the first 5 bytes with a jump to myFunc
  myFuncAddress = &myFunc;

  // will jump from the next instruction (after our 5 byte jmp instruction)
  src = (DWORD)hookedAddress + 5;
  dst = (DWORD)myFuncAddress;
  rOffset = (DWORD *)(dst-src);

  // \xE9 - jump instruction
  memcpy(patch, "\xE9", 1);
  memcpy(patch + 1, &rOffset, 4);

  WriteProcessMemory(GetCurrentProcess(), (LPVOID)hookedAddress, patch, 5, NULL);

}

int main() {
  // call original
  WinExec("notepad", SW_SHOWDEFAULT);

  // install hook
  setMySuperHook();

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

}
```

Let’s go to compile:

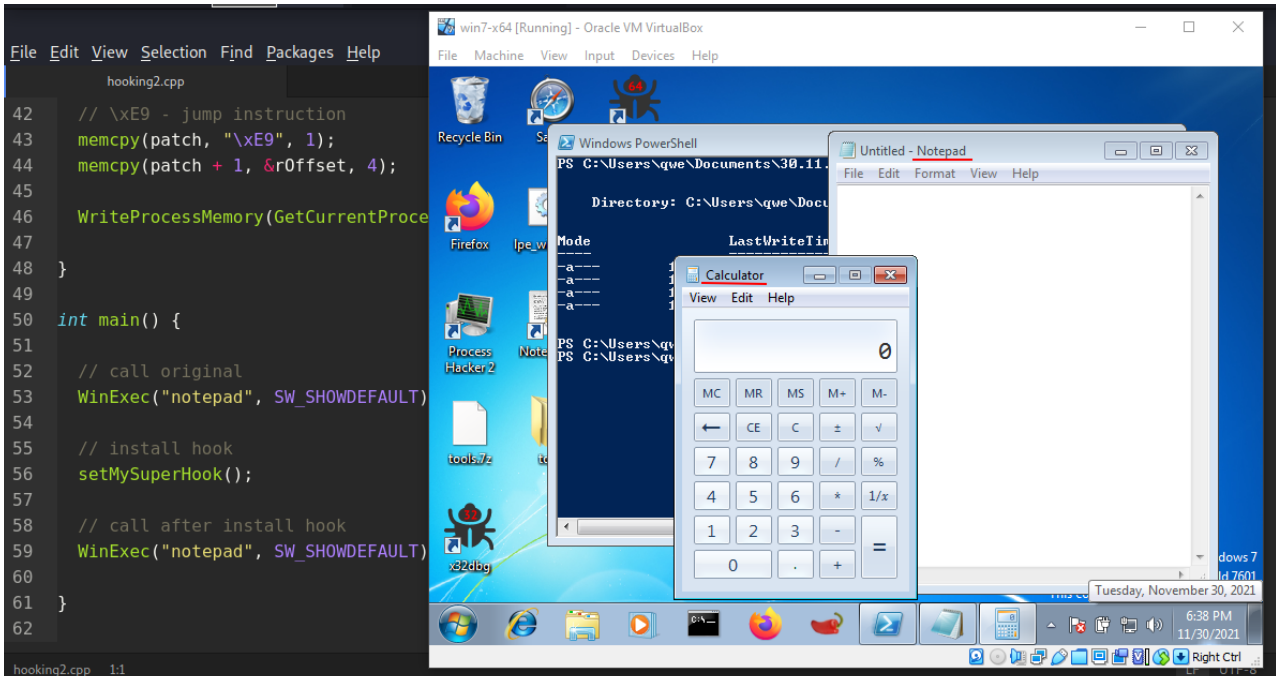
```
x86_64-w64-mingw32-g++ -O2 hooking2.cpp -o hooking2.exe -mconsole -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```


kali@kali:~/projects/cybersec_blog/2021-11-30-basic-hooking-1
File Actions Edit View Help
kali@kali:~/pr…asic-hooking-1
mc [kali@kali]…asic-hooking-1
kali@kali:~
kali@kali:~/pr…/cybersec_blog

```
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  x86_64-w64-mingw32-g++ -O2 hooking2.cpp -o hooking2.exe -mconsol
e -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -
static-libstdc++ -static-libgcc -fpermissive
hooking2.cpp: In function ‘void setMySuperHook()’:
hooking2.cpp:35:19: warning: invalid conversion from ‘int (*)(LPCSTR, UINT)’ {aka ‘int (*)(const char*, unsigned int)’} to ‘void*’
[-fpermissive]
   35 |     myFuncAddress = &myFunc;
      |                     ^~~~~~~
      |                     |
      |                     int (*)(LPCSTR, UINT) {aka int (*)(const char*, unsigned int)}
hooking2.cpp:38:9: warning: cast from ‘FARPROC’ {aka ‘long long int (*)()’} to ‘DWORD’ {aka ‘long unsigned int’} loses precision [-
fpermissive]
   38 |     src = (DWORD)hookedAddress + 5;
      |           ^~~~~~~~~~~~~~~~~~~~
hooking2.cpp:39:9: warning: cast from ‘void*’ to ‘DWORD’ {aka ‘long unsigned int’} loses precision [-fpermissive]
   39 |     dst = (DWORD)myFuncAddress;
      |           ^~~~~~~~~~~~~~~~~~~~
hooking2.cpp:40:13: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   40 |     rOffset = (DWORD *)(dst-src);
      |               ^~~~~~~~~~~~~~~~~~
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1  ls -lt
total 156
-rwxr-xr-x 1 kali kali 15360 Nov 30 18:34 hooking2.exe
-rw-r--r-- 1 kali kali  1503 Nov 30 18:34 hooking2.cpp
-rwxr-xr-x 1 kali kali 15360 Nov 30 18:22 hooking.exe
-rwxr-xr-x 1 kali kali 14336 Nov 30 18:00 cat.exe
-rwxr-xr-x 1 kali kali 92874 Nov 30 17:59 pet.dll
-rw-r--r-- 1 kali kali  1909 Nov 30 17:58 hooking.cpp
-rw-r--r-- 1 kali kali   586 Nov 30 17:52 cat.cpp
-rw-r--r-- 1 kali kali   900 Nov 30 17:51 pet.cpp
 kali@kali  ~/projects/cybersec_blog/2021-11-30-basic-hooking-1 
```

and run:

```
.\hooking2.exe
```



```
  // \xE9 - jump instruction
  memcpy(patch, "\xE9", 1);
  memcpy(patch + 1, &rOffset, 4);

  WriteProcessMemory(GetCurrentProce

}

int main() {

  // call original
  WinExec("notepad", SW_SHOWDEFAULT)

  // install hook
  setMySuperHook();

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

}
```

```
PS C:\Users\qwe\Documents\30.11.

    Directory: C:\Users\qwe\Docu

Mode                LastWriteTi
----                -----------
-a---
-a---
-a---
-a---

PS C:\Users\qw
PS C:\Users\qw
```

File Edit View Selection Find Packages Help
hooking2.cpp
win7-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Recycle Bin
Firefox
Process Hacker 2
tools.7z
x32dbg
Windows PowerShell
Untitled - Notepad
File Edit Format View Help
Calculator
View Edit Help
Tuesday, November 30, 2021
6:38 PM
11/30/2021
Right Ctrl
hooking2.cpp 1:1

So everything worked as expected.

Source code in Github

MessageBox
WinExec
Exporting from DLL using __declspec

> 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