Class 9

CSC 471 Modern Malware Analysis
Windows Message Hooks and API Hooks

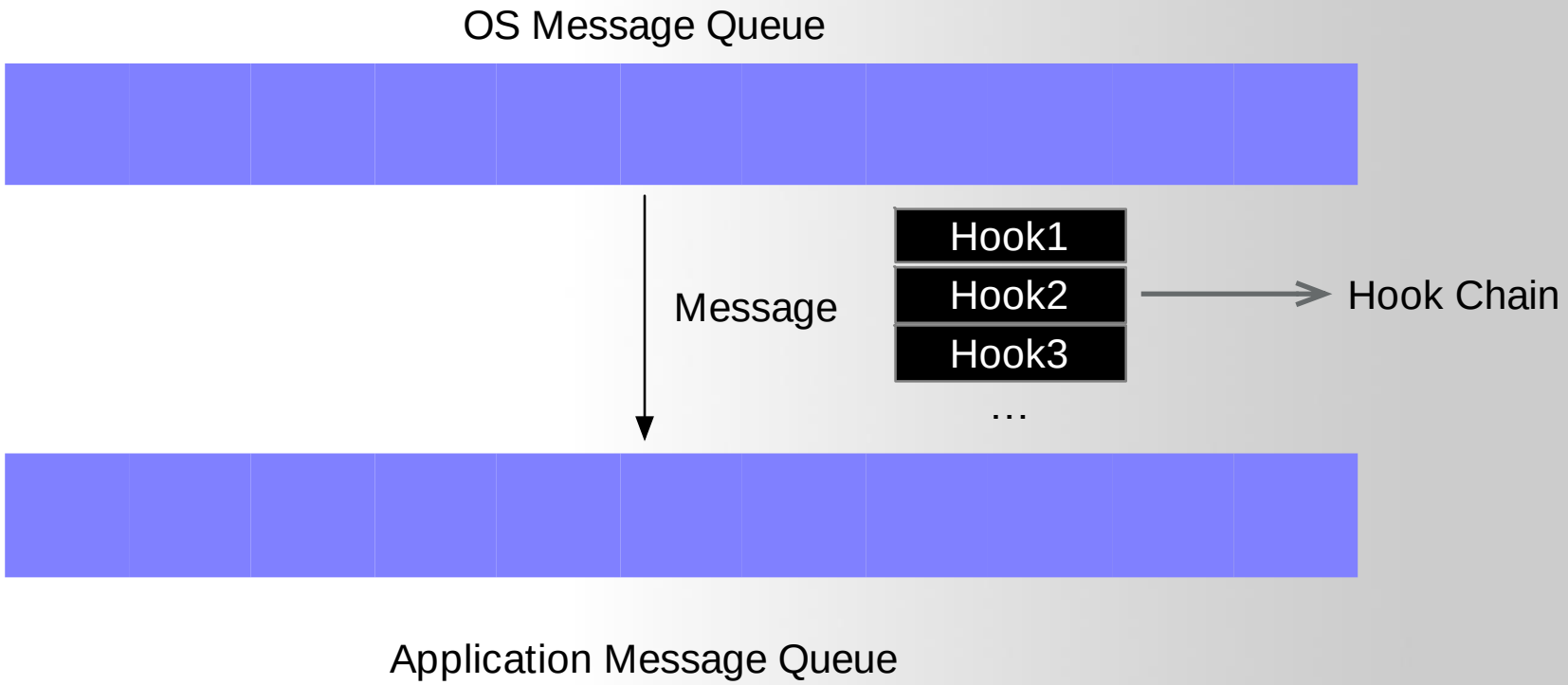
Si Chen (schen@wcupa.edu)

Message Hooks

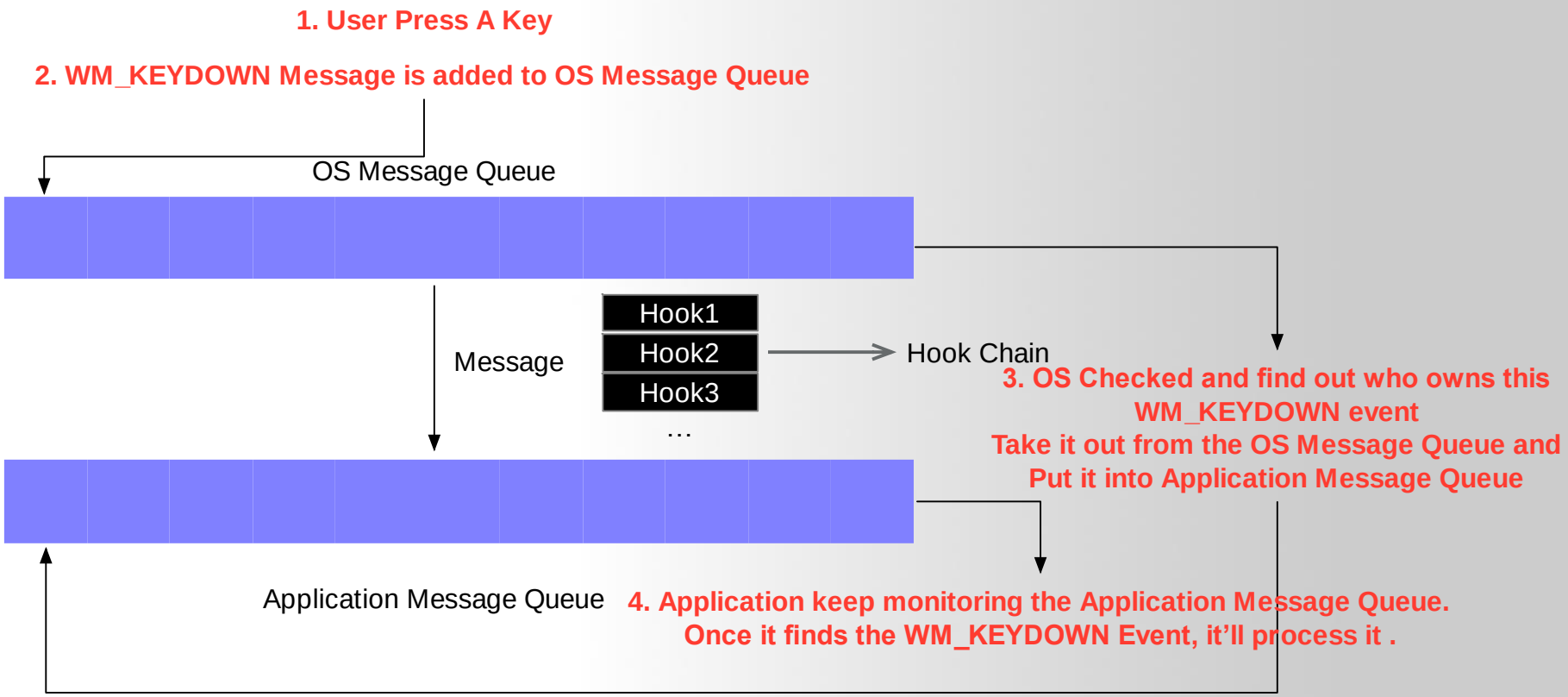
Message Hook Example

- Try HookMain.exe
- Download Hook.zip from our course website, unzip it (password: infected)

Message Hook



Message Hook

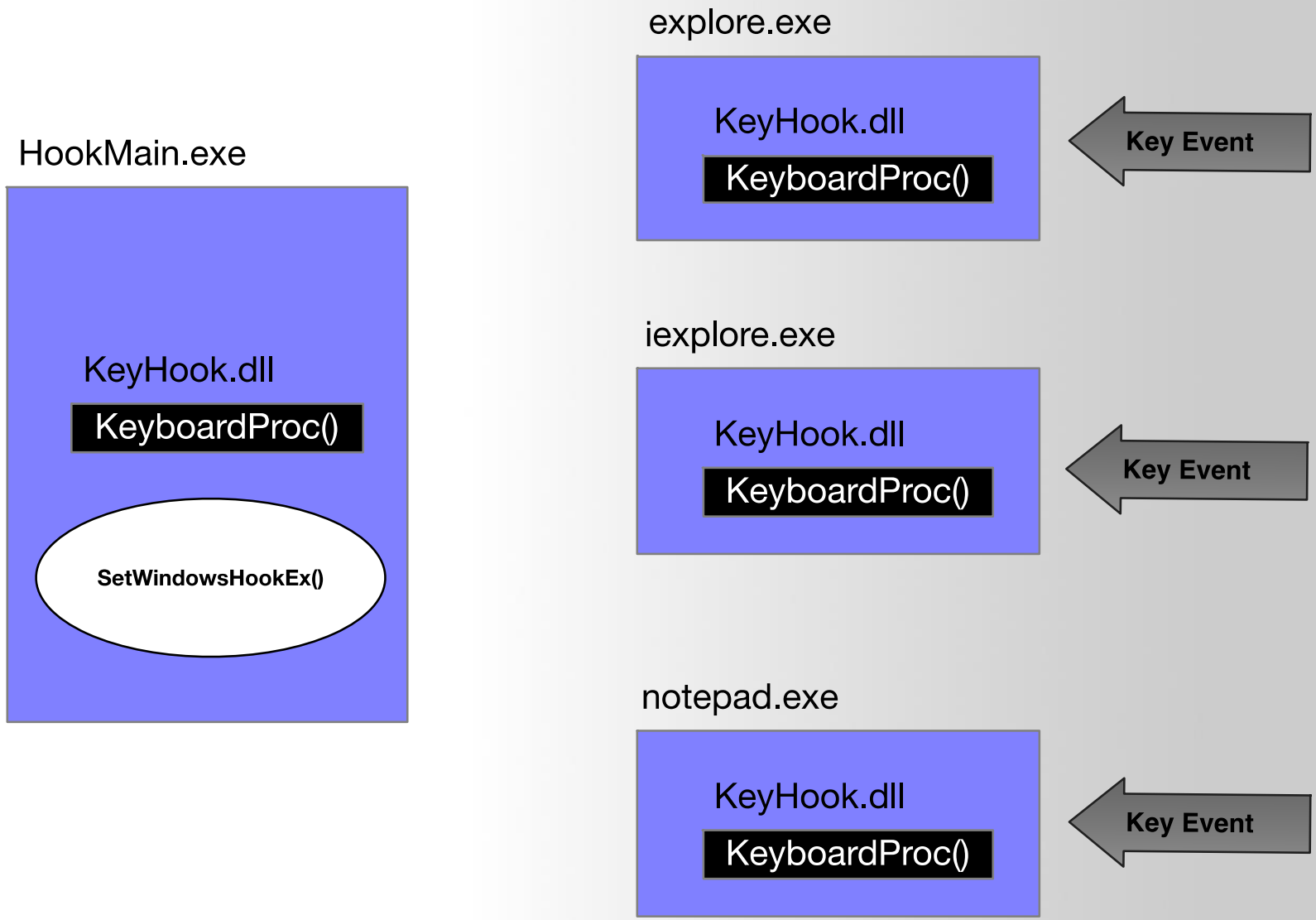


HookMain.cpp

```cpp
#include "stdio.h"
#include "conio.h"
#include "windows.h"

#define DEF_DLL_NAME        "KeyHook.dll"
#define DEF_HOOKSTART       "HookStart"
#define DEF_HOOKSTOP        "HookStop"

typedef void (*PFN_HOOKSTART)();
typedef void (*PFN_HOOKSTOP)();

void main()
{
    HMODULE         hDll = NULL;
    PFN_HOOKSTART   HookStart = NULL;
    PFN_HOOKSTOP    HookStop = NULL;
    char            ch = 0;

    // Load KeyHook.dll
    hDll = LoadLibraryA(DEF_DLL_NAME);
    if( hDll == NULL )
    {
        printf("LoadLibrary(%s) failed!!! [%d]", DEF_DLL_NAME, GetLastError());
        return;
    }

    // read export function from DLL
    HookStart = (PFN_HOOKSTART)GetProcAddress(hDll, DEF_HOOKSTART);
    HookStop = (PFN_HOOKSTOP)GetProcAddress(hDll, DEF_HOOKSTOP);

    // Start Hook
    HookStart();

    // Read user input if pressed 'q' then quit
    printf("press 'q' to quit!\n");
    while( _getch() != 'q' )    ;

    // stop hook
    HookStop();

    // unload KeyHook.dll
    FreeLibrary(hDll);
}
```

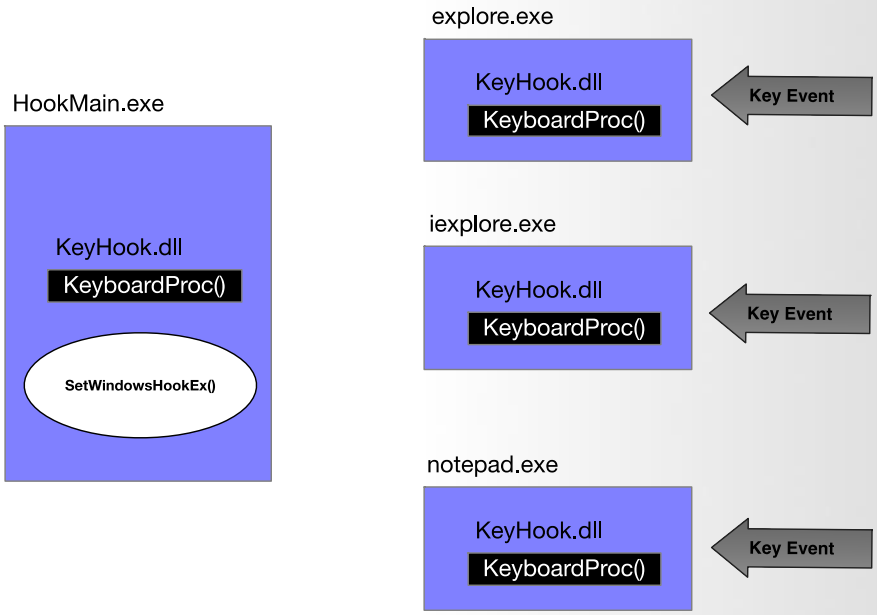
```
            // If process name is notepad.exe do not pass message
            if( !_stricmp(p + 1, DEF_PROCESS_NAME) )
                return 1;
        }
    }

    // Otherwise pass the message
    return CallNextHookEx(g_hHook, nCode, wParam, lParam);
}

#ifdef __cplusplus
extern "C" {
#endif
    __declspec(dllexport) void HookStart()
    {
        g_hHook = SetWindowsHookEx(WH_KEYBOARD, KeyboardProc, g_hInstance, 0);
    }

    __declspec(dllexport) void HookStop()
    {
        if( g_hHook )
        {
            UnhookWindowsHookEx(g_hHook);
            g_hHook = NULL;
        }
    }
#ifdef __cplusplus
}
#endif
```

Review – Message Hook



API Hook Tech Map

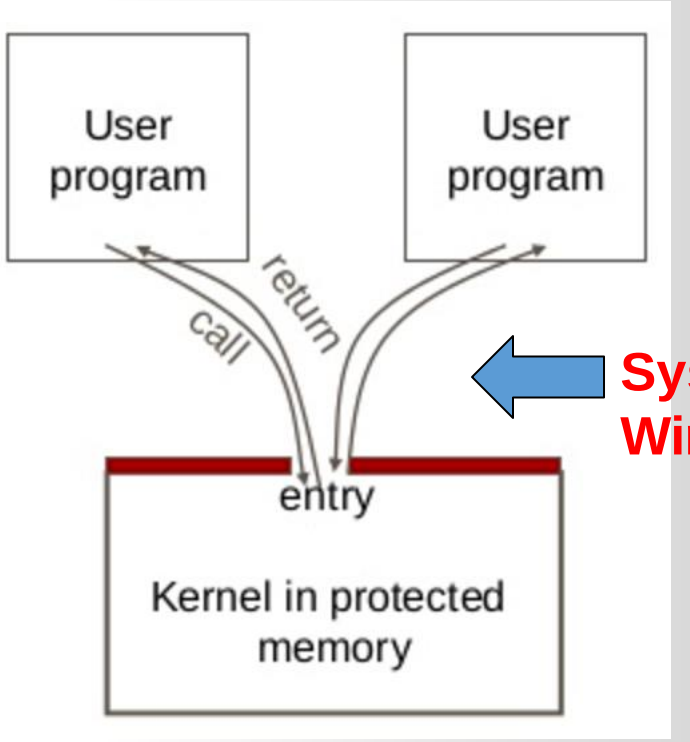
Method	Target	Location	Tech		API
Dynamic	Process/Memory 00000000 - 7FFFFFFF	1) IAT 2) Code 3) EAT	Interactive Debug		DebugActiveProcess GetThreadContext SetThreadContext
			Standalone Injection	Independent Code	CreateRemoteThread
				Dll File	Resistry (AppInit_DLLs) BHO (IE only)
					SetWindowsHookEx CreateRemoteThread



API Hooks

System Call & WinAPI

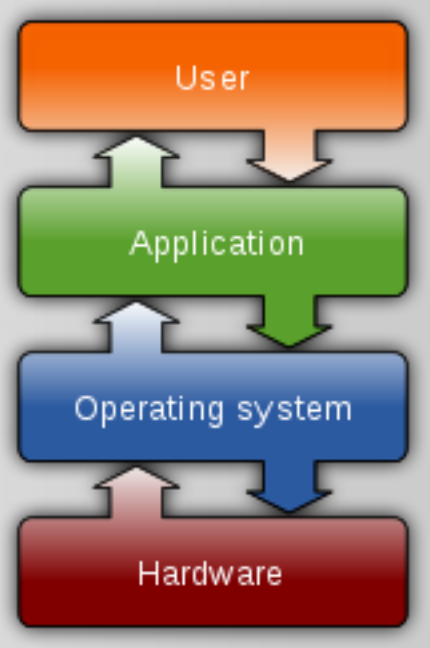
- User code can be arbitrary
- User code cannot modify kernel memory
- The call mechanism switches code to kernel mode



System Call → WinAPI (Windows)

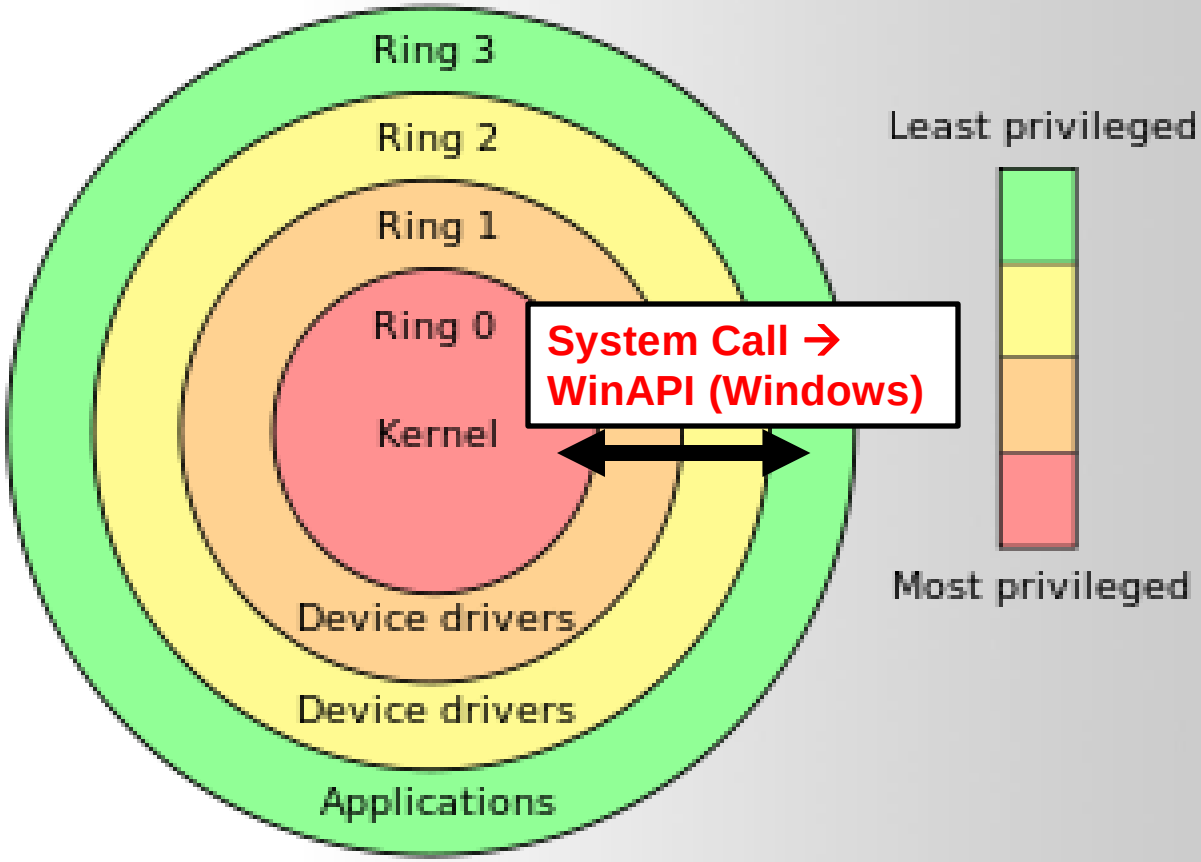
What is System Call?

- Let an application to access system resources.
- OS provide an interface (**System call**) for the application
- It usually use the technique called “interrupt vector”
  - Linux use **0x80**
  - Windows use **0x2E**

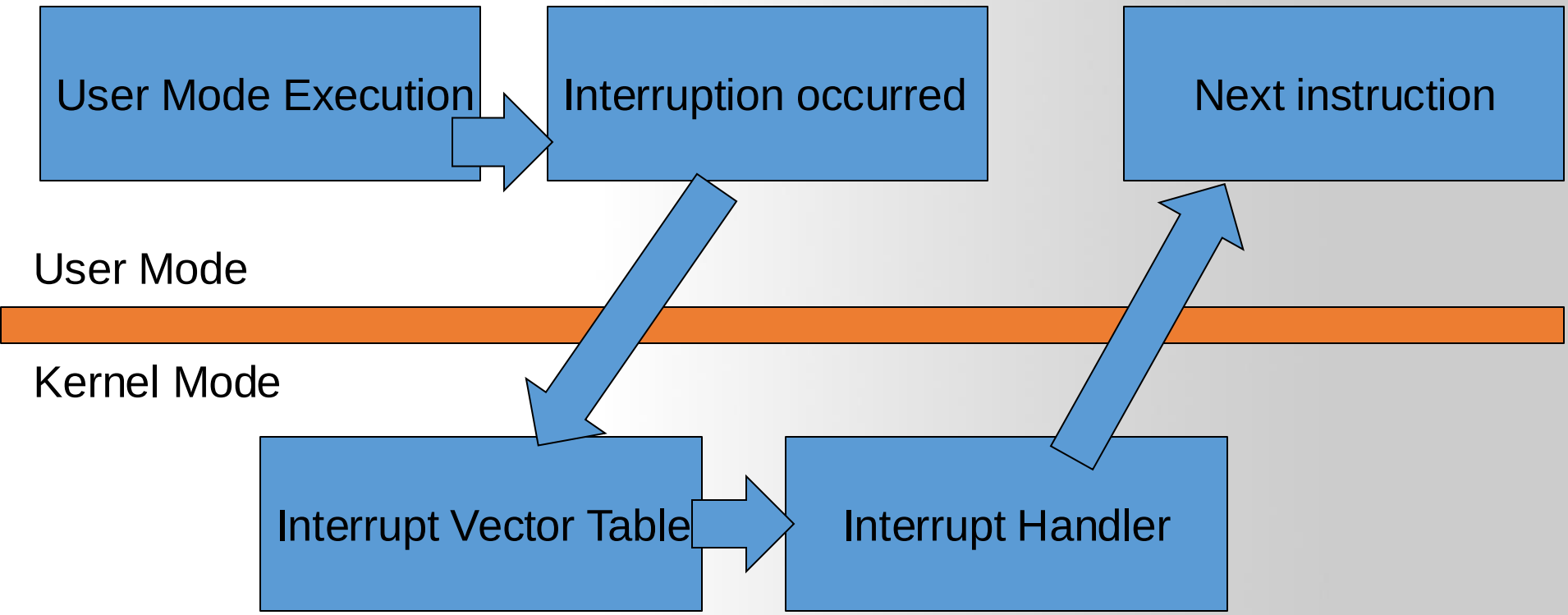


In system programming, an **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

The “Ring”

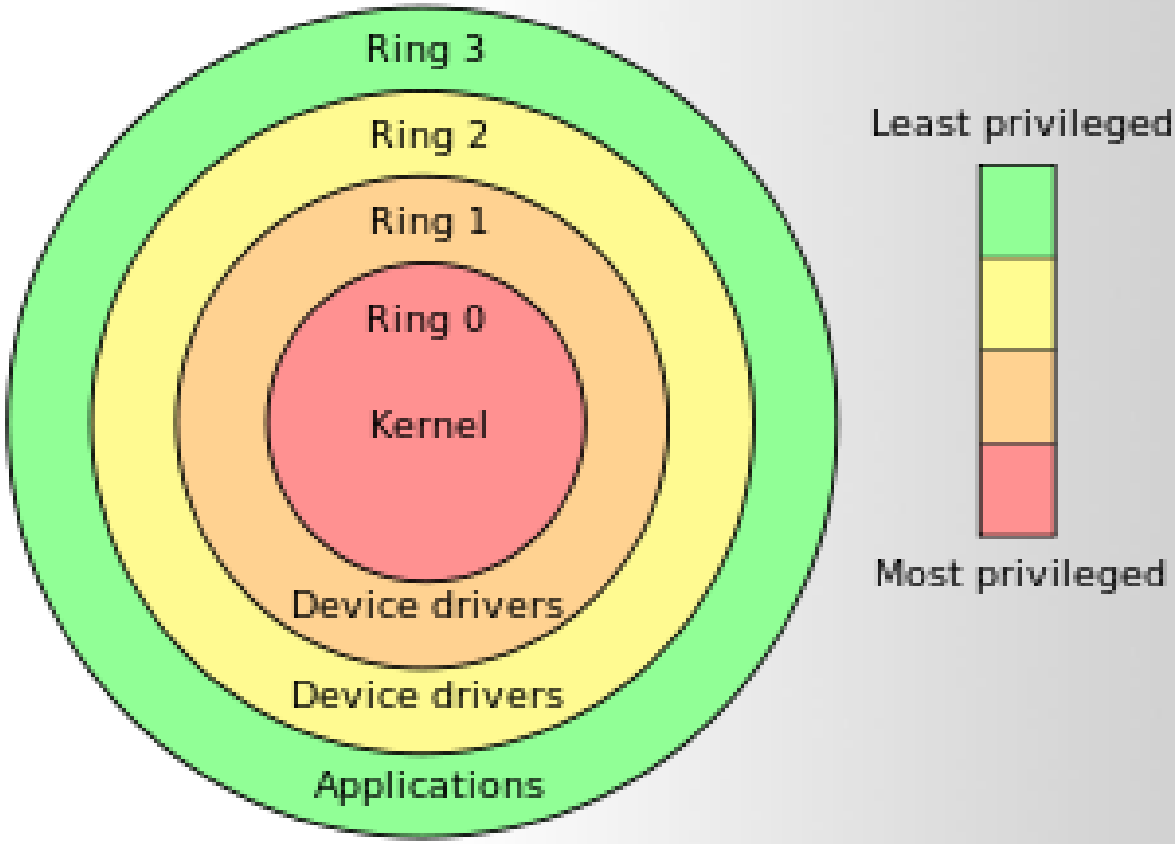

Ring 3
Ring 2
Ring 1
Ring 0
Kernel
Device drivers
Device drivers
Applications
System Call →
WinAPI (Windows)
Least privileged
Most privileged

CPU Interrupt

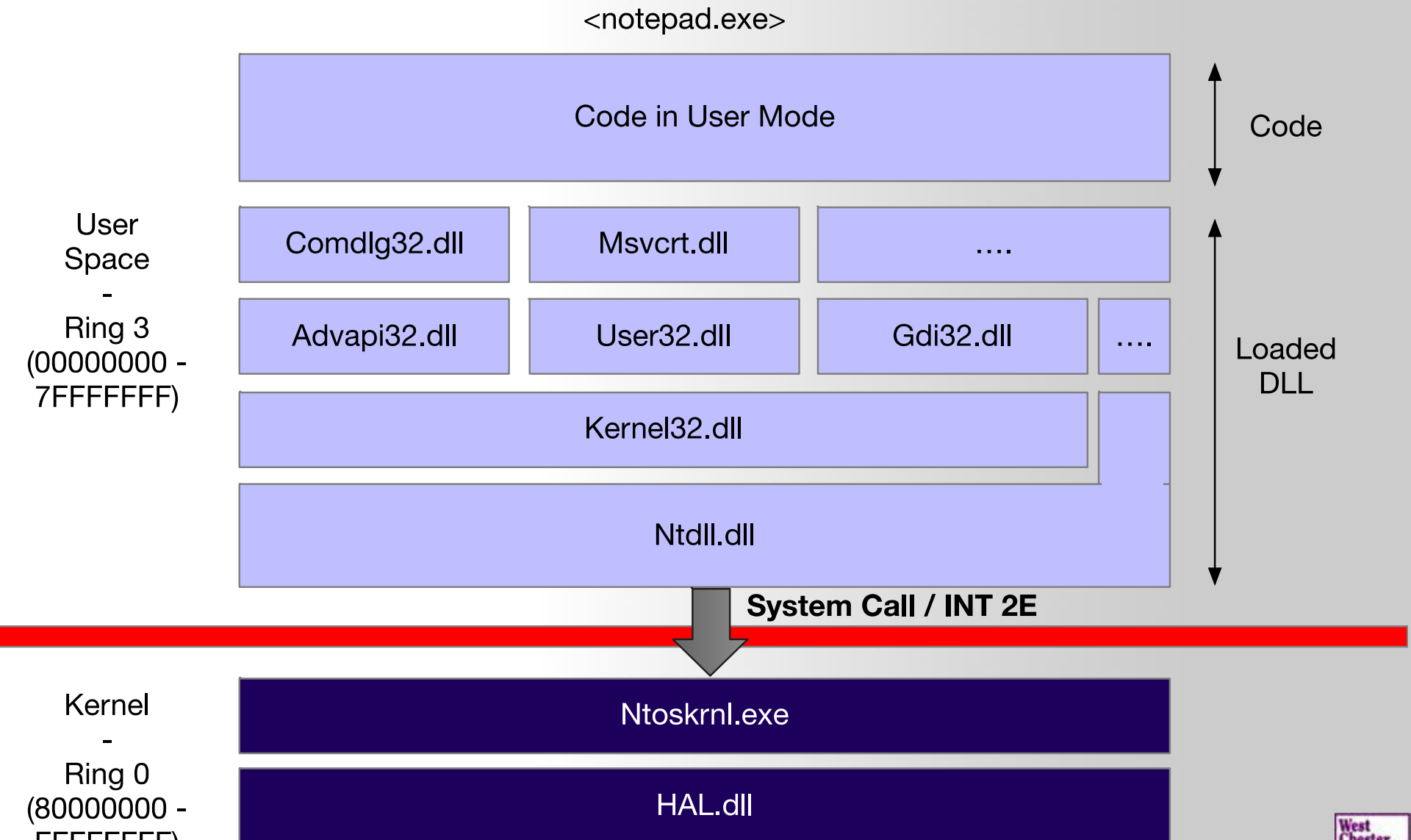


Windows System Call and API

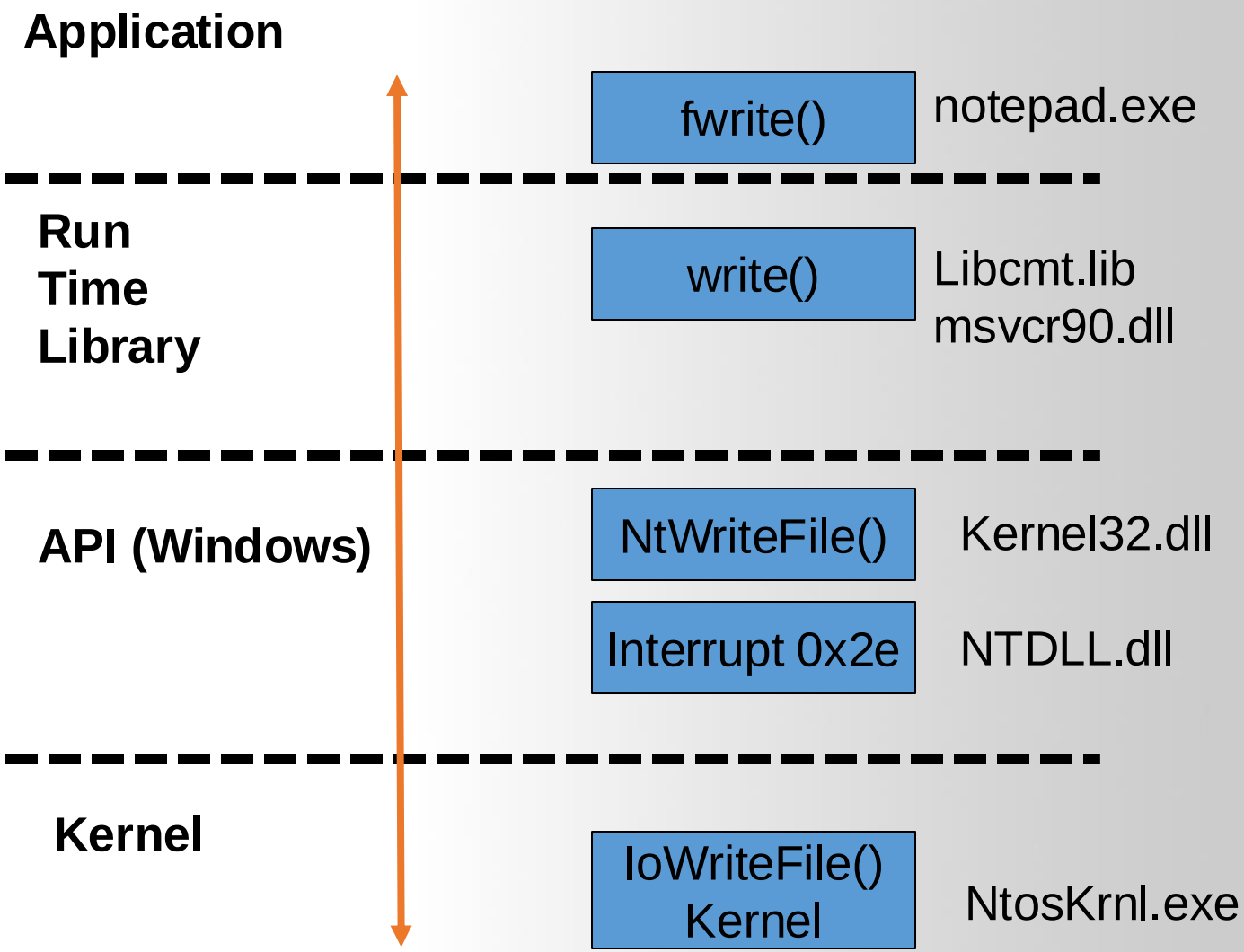
- The **Win32 API** is a layer that **runs in user mode** (ring 3).
- **Only API calls that use kernel resources** (CreateThread, VirtualAlloc, etc) will call into the "real" operating system (ntdll.dll) and trap into **ring 0** with a software interrupt (int 0x2e).



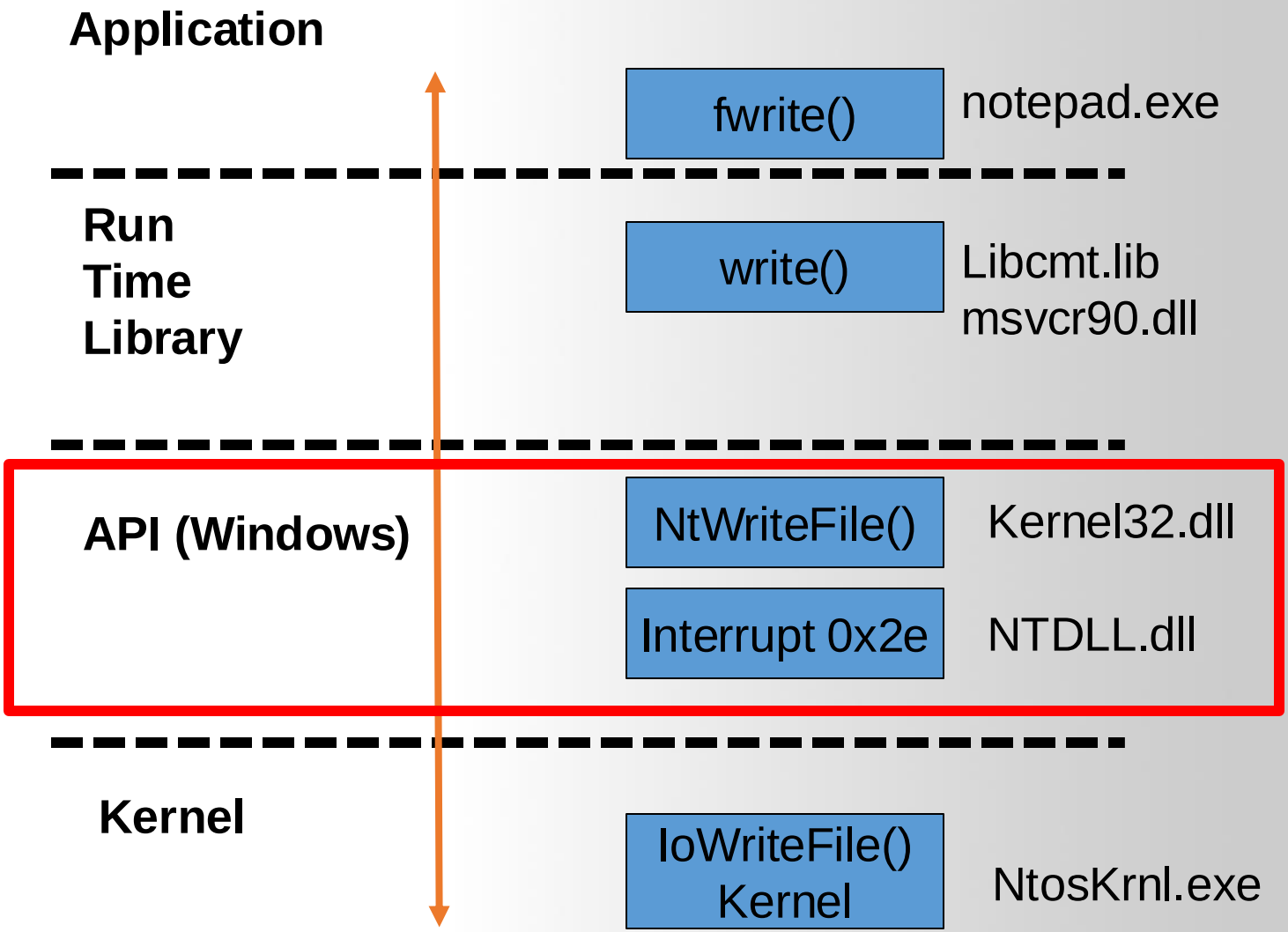
User Mode and Kernel



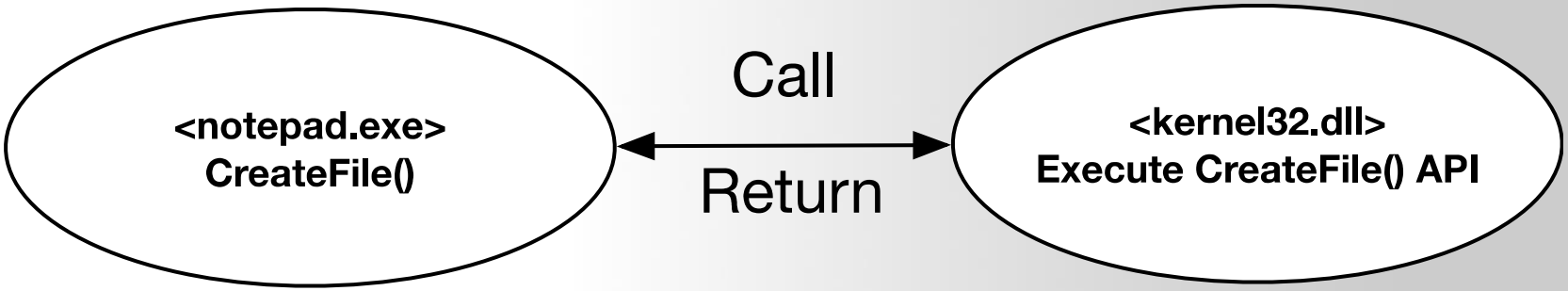
Write a file in Notepad



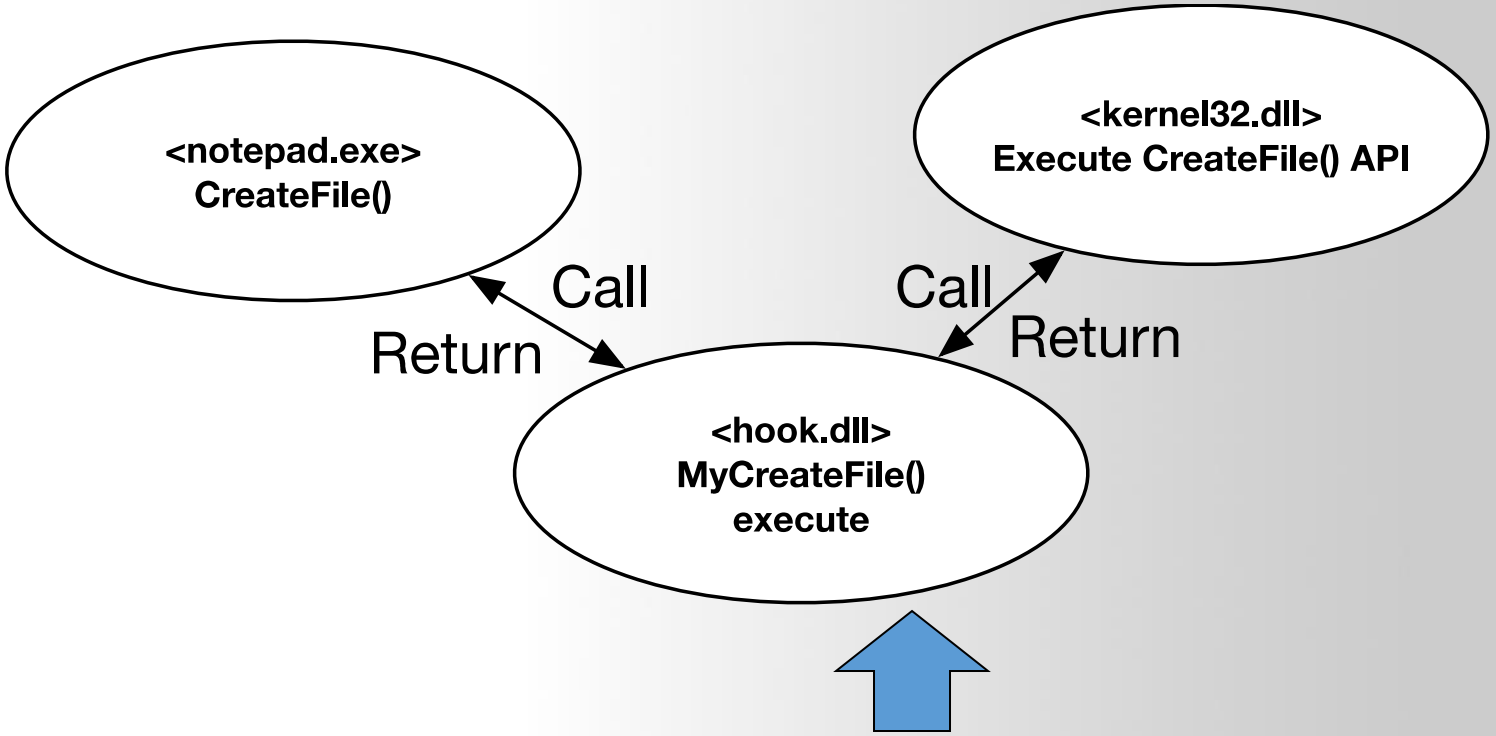
API Hook


Application
fwrite()
notepad.exe
Run
Time
Library
write()
Libcmt.lib
msvcr90.dll
API (Windows)
NtWriteFile()
Kernel32.dll
Interrupt 0x2e
NTDLL.dll
Kernel
IoWriteFile()
Kernel
NtosKrnl.exe

API Call (Normally)



API Hook

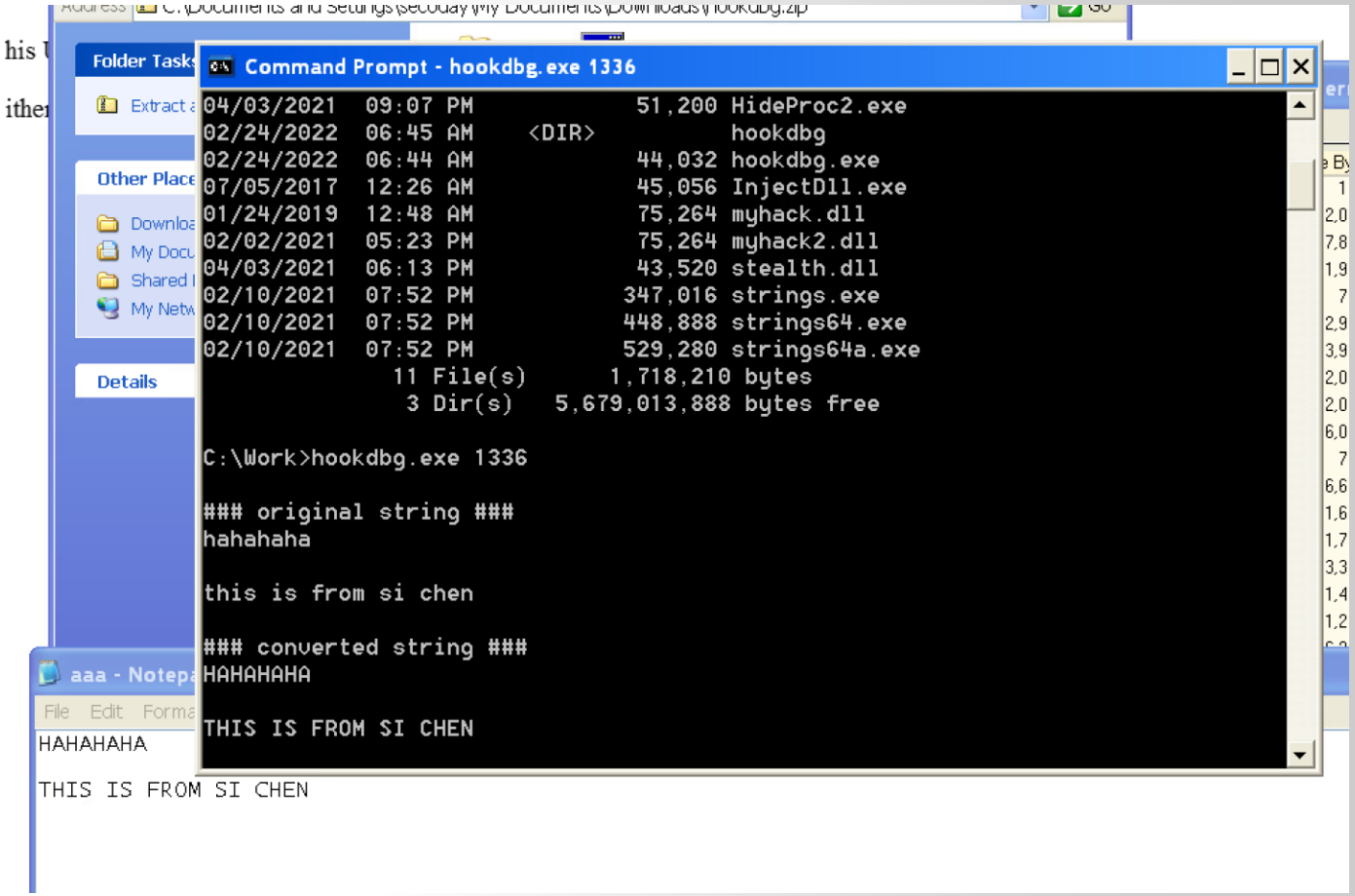

<notepad.exe>
CreateFile()
<kernel32.dll>
Execute CreateFile() API
Call
Return
Call
Return
<hook.dll>
MyCreateFile()
execute
Detour Function

API Hook Tech Map

Method	Target	Location	Tech		API
Dynamic	**Process/Memory** **00000000 - 7FFFFFFF**	**1) IAT** **2) Code** **3) EAT**	**Interactive Debug**		DebugActiveProcess GetThreadContext SetThreadContext
			Standalone Injection	**Independent Code**	CreateRemoteThread
				Dll File	Resistry (AppInit_DLLs) BHO (IE only)
					SetWindowsHookEx CreateRemoteThread

Hookdbg.exe

- API hook for Notepad WriteFile() function



Hookdbg.exe

- kernel32!WriteFile() API

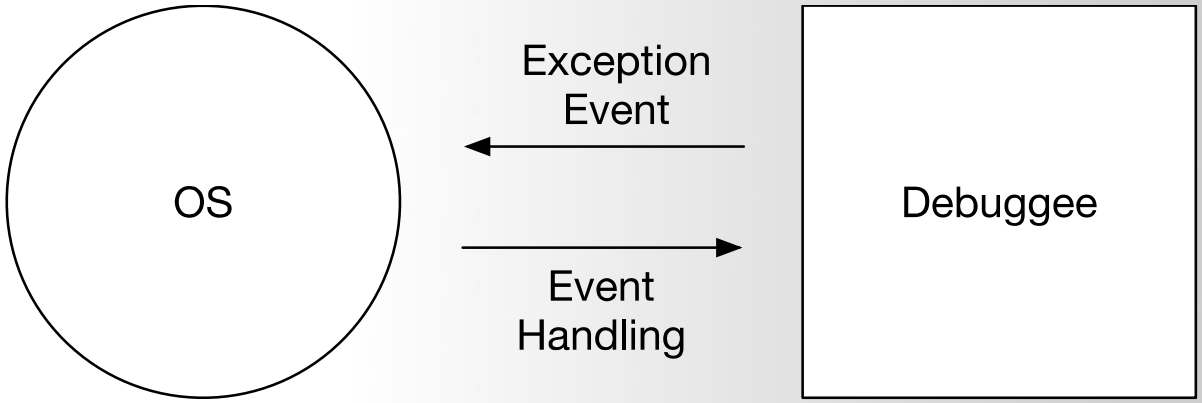
Syntax

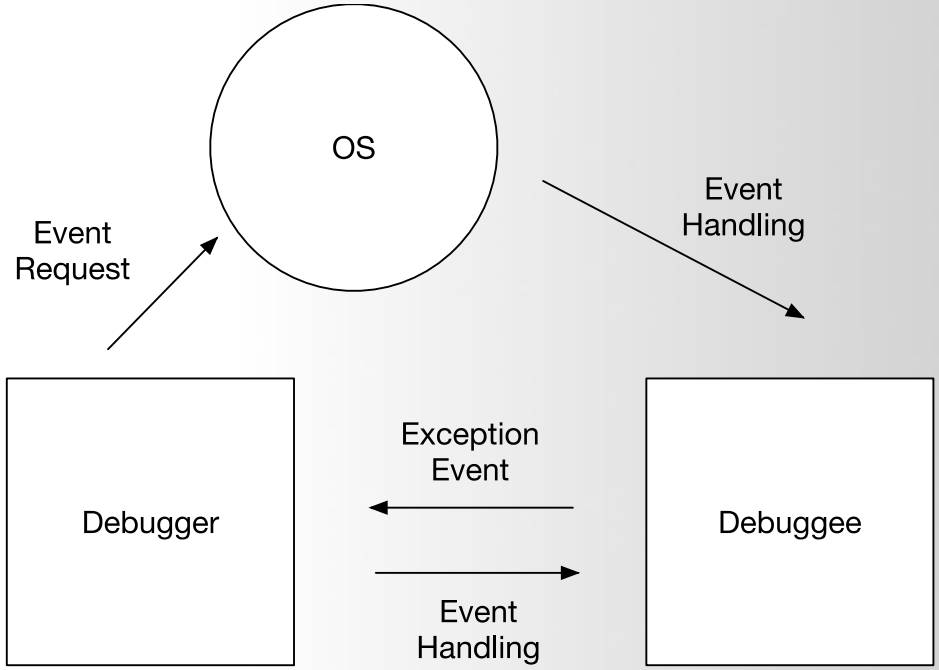
C++ Copy

```
BOOL WriteFile(
  [in]                HANDLE       hFile,
  [in]                LPCVOID      lpBuffer,
  [in]                DWORD        nNumberOfBytesToWrite,
  [out, optional]     LPDWORD      lpNumberOfBytesWritten,
  [in, out, optional] LPOVERLAPPED lpOverlapped
);
```

https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-writefile

How Debugger Works





`ExceptionCode`

The reason the exception occurred. This is the code generated by a hardware exception, or the code specified in the RaiseException function for a software-generated exception. The following tables describes the exception codes that are likely to occur due to common programming errors.

Value	Meaning
EXCEPTION_ACCESS_VIOLATION	The thread tried to read from or write to a virtual address for which it does not have the appropriate access.
EXCEPTION_ARRAY_BOUNDS_EXCEEDED	The thread tried to access an array element that is out of bounds and the underlying hardware supports bounds checking.

https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-exception_record

Q & A